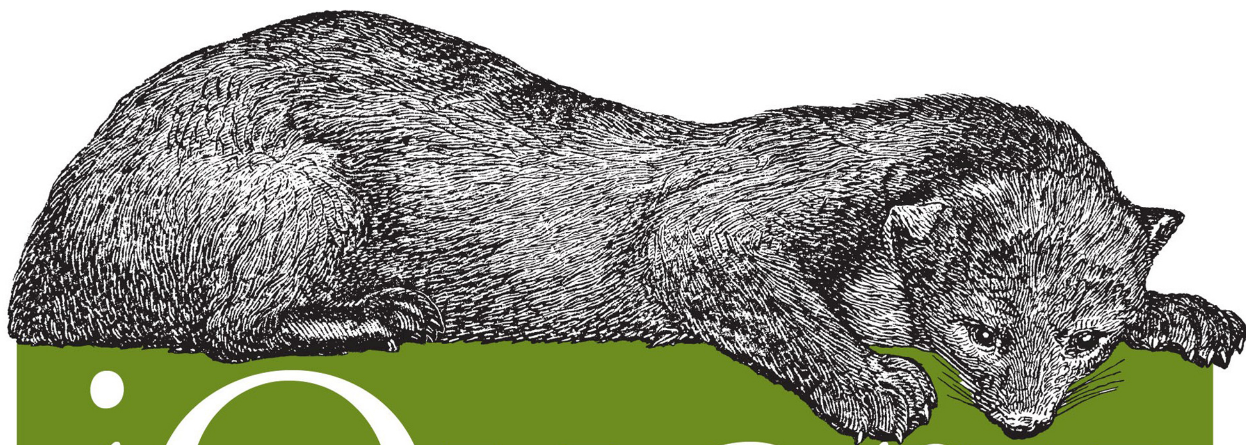
图灵程序设计丛书



jQuery Mobile 即学即用

jQuery Mobile: Up and Running

[阿根廷] *Maximiliano Firtman* 著
吴英杰 吴敏琦 译

O'REILLY®



人民邮电出版社
POSTS & TELECOM PRESS


图灵程序设计丛书

jQuery Mobile即学即用

jQuery Mobile: Up and Running

[美] Maximiliano Firtman 著
吴英杰　吴敏琦 译

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'Reilly Media, Inc.授权人民邮电出版社出版

人民邮电出版社
北　京

图书在版编目（C I P）数据

jQuery Mobile即学即用 / （美）弗特曼
(Firtman,M.）著 ; 吴英杰, 吴敏琦译. -- 北京 : 人民
邮电出版社, 2013.1
（图灵程序设计丛书）
书名原文: jQuery mobile:up and running
ISBN 978-7-115-30294-6

Ⅰ. ①j… Ⅱ. ①弗… ②吴… ③吴… Ⅲ. ①移动电
话机－应用程序－JAVA语言－程序设计 Ⅳ. ①
TN929.53②TP312

中国版本图书馆CIP数据核字(2012)第305211号

内 容 提 要

jQuery Mobile 是以全球最流行的 jQuery 为核心的、跨平台的移动 Web 应用开发框架。本书着眼于移动 Web 应用开发实战，从应用结构和导航开始，分门别类地介绍了 jQuery Mobile 为开发人员准备好的各种界面部件，包括列表视图、工具条、按钮、表单、网格布局等，还介绍了界面主题和配色，以及为 jQuery Mobile 编程准备的事件、配置及响应式布局 API。本书最后向读者展示了一个完整的 Web 应用开发示例。

本书适合所有对开发移动 Web 应用感兴趣的读者阅读参考。

图灵程序设计丛书

jQuery Mobile即学即用

◆ 著　　[美] Maximiliano Firtman
　译　　吴英杰　吴敏琦
　责任编辑　李松峰
◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
　邮编　100061　　电子邮件　315@ptpress.com.cn
　网址　http://www.ptpress.com.cn
　北京　　　　印刷
◆ 开本：800×1000　1/16
　印张：15
　字数：355千字　　　2013年 1 月第 1 版
　印数：1 – 3 500册　　　2013年 1 月北京第 1次印刷

著作权合同登记号　图字：01-2012-8792号

ISBN 978-7-115-30294-6

定价：49.00元

读者服务热线：(010)51095186转604　印装质量热线：(010)67129223

反盗版热线：(010)67171154

版权声明

O’Reilly Media, Inc.介绍

O’Reilly Media 通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自 1978 年开始，O’Reilly 一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O’Reilly 的发展充满了对创新的倡导、创造和发扬光大。

O’Reilly 为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了 Make 杂志，从而成为 DIY 革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O’Reilly 的会议和峰会聚集了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O’Reilly 现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版、在线服务或者面授课程，每一项 O’Reilly 的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O’Reilly Radar 博客有口皆碑。”

——Wired

“O’Reilly 凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 2.0

“O’Reilly Conference 是聚集关键思想领袖的绝对典范。”

——CRN

“一本 O’Reilly 的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim 是位特立独行的商人，他不光放眼于最长远、最广阔的视野并且切实地按照 Yogi Berra 的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去，Tim 似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

目录

前言

如果你是一名 Web 设计师或开发者，并且正在考虑使用 jQuery Mobile 创建移动应用，那么本书将是你理想的伙伴。

jQuery Mobile 致力于解决一个问题：为市场上众多移动平台和浏览器创建兼容界面。

要看懂本书，你只需要了解基本的 HTML（任何版本）知识，而懂一些基本的 JavaScript 则将对理解后面章节的内容非常有帮助。读者不需要会 HTML5、JavaScript 或 jQuery 即可使用 jQuery Mobile 框架或本书的大部分内容。

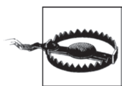
本书排版约定

本书使用以下排版约定。

- 楷体
  用于标识新术语。
- 等宽字体
  用于显示程序，也用于段落中的代码元素，如变量或函数名、数据库、数据类型、环境变量、语句以及关键字。
- 等宽粗体
  用于显示那些需要用户原样输入的命令或文本。
- 等宽斜体
  用于显示需要用户输入或取决于上下文的文本。



这个图标表示提示、建议或一般注解。



这个图标表示警告或注意。

使用代码

本书旨在帮助你完成工作。一般来说，你可以在你的程序及文档中使用本书中的代码。除非你准备复制大块的代码，否则无需从我们这儿获得许可。例如，写一个用到几段本书中代码的程序时不需要得到许可，而出售或传播 O'Reilly 书籍中示例的光盘（CD-ROM）则需要许可，引用本书内容或书中的示例代码回答问题无需得到许可，将本书中的大段示例代码编入你的产品文档需要得到许可。

我们赞赏但不强求注明信息来源。信息来源通常包括标题、作者、出版商以及国际标准书号（ISBN）。如：" *jQuery Mobile: Up and Running* by Maximiliano Firtman (O'Reilly). Copyright 2012 Maximiliano Firtman, 978-1-449-39765-4."。

如果你觉得对示例代码的使用超出了正当引用或上面给出的许可范围，请随时通过 permissions@oreilly.com 联系我们。

Safari®在线图书



Safari 图书在线是一个数字图书馆，可以供你方便快捷地在 7500 多种技术和创意类参考书以及视频中找到需要的信息。

订阅之后，就可以在这个图书馆在线阅读、查看所有图书及视频。你可以在移动设备上阅读，在新主题付印之前就阅读它们，还可以查看编写中的手稿并直接给作者反馈、复制粘贴代码、管理收藏、下载章节、给关键段落加书签、创建批注、打印页面，以及获得其他众多节约时间的好处。

O'Reilly Media 已经将本书（英文版）上传至 Safari 图书在线。要获得本书及其他来自 O'Reilly 等出版商类似选题的电子版完全访问权限，请到 http://my.safaribooksonline.com 免费注册。

我们的联系方式

如果想对本书发表评论或有任何疑问，请联系出版社。

美国：

O’Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）
奥莱利技术咨询（北京）有限公司

O’Reilly 的每一本书都有专属网页，你可以在那儿找到本书的相关信息，包括勘误表、示例代码以及其他信息。本书的网站地址是[1]：

http://shop.oreilly.com/product/0636920014607.do

中文书：

http://www.oreilly.com.cn/index.php?func=book&isbn=978-7-115-30294-6

对于本书的评论和技术性问题，请发送电子邮件到：

bookquestions@oreilly.com

关于本书的更多信息、会议、资源中心和网络，请访问以下网站：

http://www.oreilly.com
http://www.oreilly.com.cn

我们在 Facebook 的地址如下：

http://facebook.com/oreilly

请关注我们的 Twitter 动态：

http://twitter.com/oreillymedia

我们的 YouTube 视频地址如下：

http://www.youtube.com/oreillymedia

编注 1：你也可在图灵社区（www.ituring.com.cn）本书网页下查看中文版勘误、评论和相关文章。

第 1 章 移动平台

如果你正在阅读本书，说明你可能是一位网页设计师或网页开发者，也可能是一位 jQuery 爱好者或一位 Web 应用开发者。在开始写代码之前，我们首先要了解一下移动生态系统以及 jQuery Mobile 适合做些什么。让我们开始吧。

1.1 为什么需要jQuery Mobile

你可能会问的第一个问题是 jQuery Mobile 存在的意义是什么？既然已经有大量能渲染标准桌面网站的移动浏览器了，为什么还要专门考虑移动设备？

让我从我的另一本书《移动网络程序设计》（*Programming the Mobile Web*）中复制粘贴一些内容来回答这些问题。当然，我已经从自己这儿得到授权了。

1.1.1 移动互联网的传说

随着人们访问互联网的方式延伸到移动终端，开发者中间也流传着很多故事，这些故事讲述了这个趋势将给他们的工作带来什么影响。虽然有些故事是真实准确的，但也不乏带来误导的、让人迷惑的，甚至是极端错误的描述。

1. 没有所谓的移动互联网，只有一个互联网

过去的几年里我曾经多次听说这个言论。的确只有一个互联网，想一下你的生活吧，你不会为你的手机专门设置另一个 Email 账号。（好吧，我知道有些人会这样做，但我相信这并不是普遍情况。）

在最喜欢的网站上，比如在 ESPN 上阅读最新的 NBA 赛事的消息，台式机和手机访问的都是同一个新闻源。你也真的不想用手机访问另一个社交网络，而是希望使用和桌面电脑上相同的 Facebook 或 Twitter 账号。在桌面电脑上创建朋友列表已经很痛苦了，你已经因此忽略了很多人，你可不想把这些事在手机上再重做一次。

因此，只有一个互联网。不过，在为移动互联网做开发时，我们面对的是非常、非常不一样的设备。最明显的不同是屏幕尺寸，这也将是我们遇到的第一个问题。除此之外，还有很多不那么明显的差异，比如我们使用移动设备的情境，通常和使用舒适的桌面电脑甚至笔记本、上网本时的地点、场景完全不同。

不要误会我，这不是说，开发者必须为我们的工作创建两个、三个或多个版本。这就是 jQuery Mobile 要解决的问题。

2. 设计移动网站不需要什么特别处理

今天市场上几乎所有的智能手机，如 iPhone 以及基于 Android 的设备，都可以读取并显示完整的桌面站点，这是事实。用户希望在移动互联网上的体检和在桌面电脑上的一致，这也是事实。一些统计甚至显示，用户使用智能手机时，更倾向于选择桌面版本而不是移动版本。不过，这是因为我们喜欢通过缩放、滚动以及旋转操作来浏览信息呢，还是因为移动版本的用户体验实在太差了？我曾经见过很多移动互联网网站，除了 logo 和若干文本链接之外什么也没有。我希望在智能手机上看到更多的东西！

3. 一个网站应当在所有设备（台式机、手机、电视）上都能运转

如同我们即将看到的，有一些技术能让我们只创建一个文件就能为包括桌面电脑、手机、电视以及游戏终端在内的众多设备提供不同的体验，这个愿景被称为 One Web。今天，有很多移动设备的连接速度很慢，资源也受限——我说的是非智能手机，理论上它们也能读取并解析文件，但如果我们把为桌面设备准备的文档发给它们，则它们非但不能提供最好的体验，还可能产生一些兼容性问题。因此，One Web 仍然是将来时。为了让不同的移动设备都能提供良好的用户体验，还需要做一些额外工作，不过也有一些技术可用于减少这些工作并避免代码和数据的重复。

4. 只要创建一个240像素宽的HTML文件，就算是一个移动网站了

这是另一种关于移动互联网的简单化认识。今天，市场上有 3000 多种移动设备，差不多有 50 种不同的浏览器（事实上，如果我们按它们的版本号来划分的话，有 500 多种不同的浏览器）。只有一个 HTML 文件的移动网站是相当不靠谱的。另外，这种做法，实际上是在助长移动互联网毫无用处的错误观念。

1.1.2 移动Web应用

我并不打算讨论移动 Web 开发与原生开发孰优孰劣，事实上，我相信这个讨论本身没有多大意义。通常，这些讨论的焦点就是对比原生代码与 JavaScript 代码，或浏览器应用与本地应用。不过，这些讨论可能不会提到的是，对原生开发环境来说，多平台开发是有挑战的，因为每个平台都有不同的 SDK。既然问题的核心是要便利地在多个移动设备上开发及发布，那么移动 Web 开发在大多数情况下都是一个完美的解决方案。Web 应用有着大量的同义词或近义词，如移动 Web 应用、小部件（widget）、聚合（hybrid）、HTML5 应用等。

特别是，移动 Web 应用与通常的移动网站在目标上并不相同。一个 Web 应用通常会模仿原生的手机应用，有着更加事务型的用户界面。它虽然仍是由网页技术（HTML、CSS、JavaScript、AJAX）创建的，但向用户提供了一个类似应用程序的体验。

很多移动 Web 应用也使用了离线或地理定位（geolocation）等 HTML5 特性，以便提供更好的体验。地理定位不是 HTML5 官方标准，而是 W3C 的一个独立 API，不过，它经常以 HTML5 名义被提到。

Web 应用可以用很多方式（见图 1-1）实现，包括：

- 从浏览器中访问；
- 安装为一个全屏的 Web 应用；
- 通过提供商实现的官方包（有时叫小部件）安装；
- 内嵌在本地应用中，通常称为聚合。

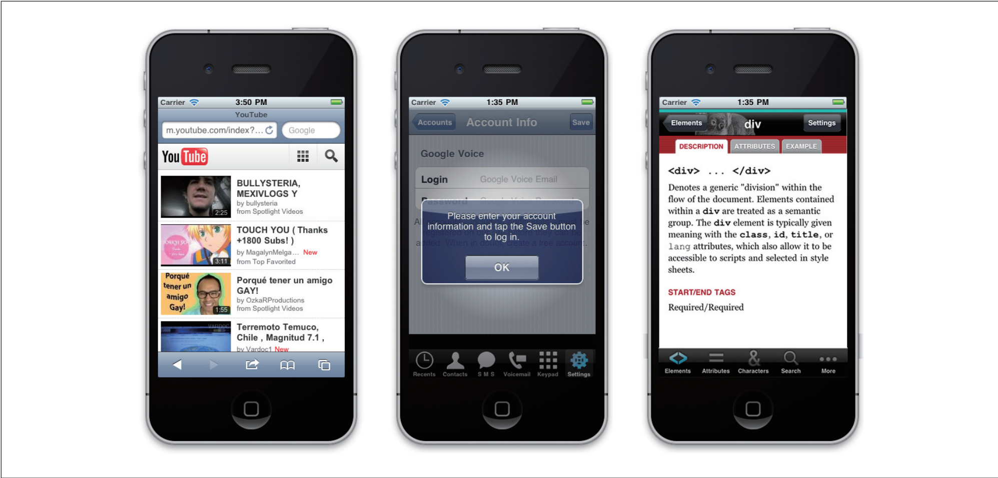


图 1-1：Web 应用（从左到右）：基于浏览器的体验、安装的全屏应用、在本地应用中嵌入的 Web 应用

本书剩下的部分将介绍如何创建这些网页应用。想了解这方面的更多信息，可以阅读我的另一本书《移动网络程序设计》。

Web 应用通常会为网页设计师和开发人员带来新的挑战，比如加载视图而不是页面，管理视图之间的双向导航，以及创建为触屏设备定制的富控件等。

1.1.3 再问一次，为什么需要jQuery Mobile

如果你已读过了上面几页（我相信你已经读过了），就会知道在移动互联网设计及开发中有一些新的挑战。我们需要创建 Web 应用，而不仅仅是简单的网站。有太多的设备以及浏览器要兼容，同时，也有许多由社区和设备提供商开发的程序库尝试解决这些问题。

这就是 jQuery Mobile 诞生的原因：让设计师和开发者使用少量代码即可更容易地创建跨平台、可定制的移动互联网体验。

jQuery 世界范围的广大社区也将为这个框架的未来提供大量机会。

这个框架同时还获得了移动领域许多大牌公司的官方赞助和支持，其中包括：

- Adobe；
- Mozilla 公司；
- HP Palm；
- BlackBerry/RIM；
- Nokia；
- DeviceAtlas 及 dotMobi。

1.2 jQuery Mobile是什么

根据位于 http://www.jquerymobile.com 的官方说明：

> jQuery Mobile 是一个支持所有流行移动设备平台的统一的用户界面系统，基于坚如磐石的 jQuery 及 jQuery UI。它轻量级的代码使用渐进增强方式构建，具有可伸缩、易更换主题的设计特点。

1.2.1 jQuery Mobile不是什么

知道 jQuery Mobile 不是什么，对理解 jQuery Mobile 十分重要。

- jQuery Mobile不是移动浏览器上的 jQuery。
  要使用 jQuery Mobile，需先载入通常的 jQuery 框架。或者说，它并不是 jQuery 的替代，而是一个构建于 jQuery 之上的 UI 层。

- jQuery Mobile不是Web应用的开发包。
  可以使用 jQuery Mobile 来创建完整的移动应用体验，但你仍然需要一些额外的工作来将它编译为原生应用。在后面几章中，我们将了解到为什么要这样做，以及具体何时、怎样来做。

- jQuery Mobile不是专属于JavaScript爱好者的框架。
  除了一些高级主题，使用 jQuery Mobile 时一般无需涉及任何 JavaScript 代码。对于讨厌大括号、分号这些 JavaScript 语法的 Web 设计师而言，这是一个好消息。

- jQuery Mobile不是面向所有移动应用/网站/游戏的解决方案。
  尽管如此，jQuery Mobile 可以为它们中的大多数提供解决方案。至于剩下的那些，我不得不建议你去阅读我写的另一本相关书籍。

1.2.2 框架

如果你没听说过 jQuery，那么你大概是从十年前穿越过来的吧？好吧，如果你是 Marty McFly[1]，请通过浏览器访问 http://jquery.com，了解一下这个超级有用的 JavaScript 框架。事实上，自 2007 年开始，它就是互联网上使用最为广泛的 JavaScript 框架了。

jQuery Mobile 是一个帮助开发者更容易地在移动设备和平板电脑上（这些设备大多具备触摸操作功能）交付跨平台 Web 应用的框架，只使用标准 HTML5 代码。一个 jQuery Mobile 应用看起来大致如图 1-2 所示。

jQuery Mobile 使用了 jQuery 核心，此外还包含一个 JavaScript 库、一个 CSS3 样式表，以及一些资源图片。

jQuery Mobile 和桌面端的 jQuery UI 类似，即它同样只是一个 UI 框架。它的名字没有包含 UI 这个词，这也许会让你怀疑它是一个核心框架。在我看来，这个命名选择应该是想借力于 jQuery 品牌在设计师和开发者圈子中的巨大影响力。



jQuery Mobile 是由 jQuery 主框架的开发团队创建的，由来自 Mozilla 的 JavaScript 工具开发者 John Resig 带队（他的 twitter 账号是 @jeresig）。

译注 1：经典科幻电影《回到未来》三部曲中的男主人公。

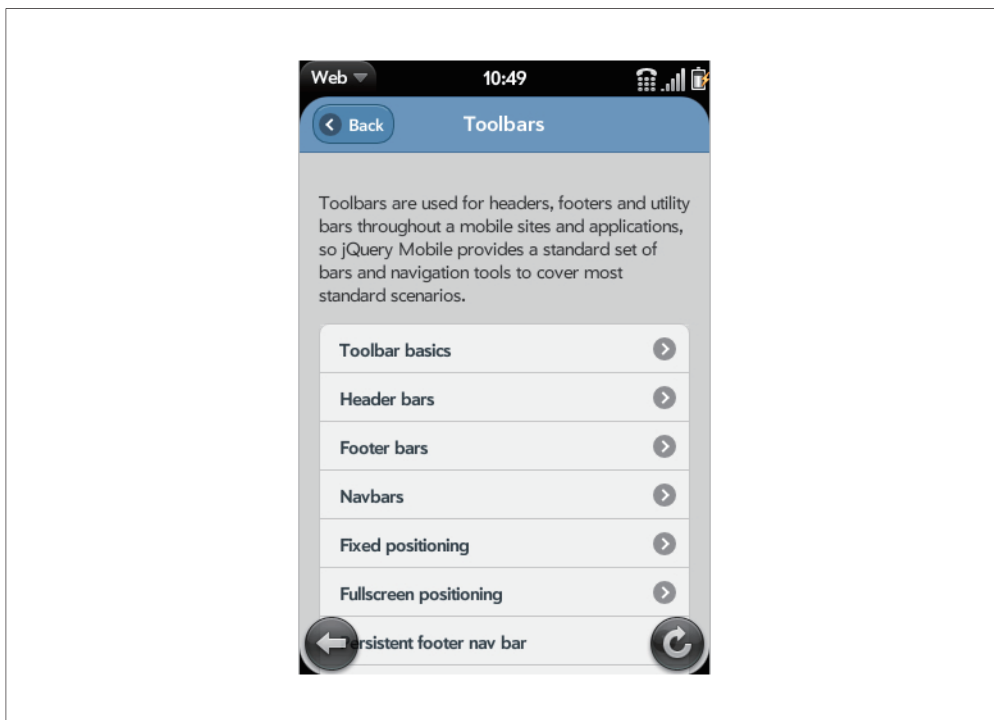


图 1-2：一个使用标准主题的典型 jQuery Mobile Web 应用，本例中使用的是 webOS 智能手机

和 jQuery 及 jQuery UI 一样，这个新的平台也是基于 MIT 或 GPLv2 双协议发布的开源项目。



如果想参与 jQuery Mobile 的开发，可以通过 http://jquerymobile.com/contribute 这个网址提供补丁、bug 修复以及参与讨论。

1.3 移动及平板的世界

人们已不再只通过桌面电脑来浏览互联网了。现在，我们有很多种设备，它们有着不同的屏幕尺寸、输入方式，甚至连我们的老朋友如 HTML、JavaScript 以及 CSS 都有了各种新的特性。

毫无疑问，移动设备的时代已经来临。全球有 50 多亿台移动终端，并且这个数目仍在增长中。与此同时，平板电脑也正迅速普及，总量以百万计。

1.3.1 设备分类

现在，我们可以将移动设备分为以下几类：

- 移动电话；
- 低端移动设备；
- 中高端移动设备，也被称为社交设备；
- 智能手机；
- 平板电脑。

1. 移动电话

是的，在某些市场上，我们还能看见移动电话的身影。它们只具备通话和短信功能，没有浏览器，不能连接互联网，无法安装应用。我们不关心这些设备，也不能为它们做任何开发。

要不了几年，随着运营商和设备制造商发起的设备回收以及产品更新服务，这些设备也许就会从市场上彻底消失了。

2. 低端移动设备

低端移动设备有一个很大的优势，即具备 Web 接入能力。它们一般只自带一个入门级的浏览器，但总量占市场的主流。也许这些用户目前还不是互联网上最活跃的用户，但随着社交网络和 Web 2.0 服务的兴起，一切可能很快会发生变化。如果你的朋友可以从移动设备上发布照片，你或许会希望自己也可以做到，所以只要条件成熟，你可能就会换一新部手机。

诺基亚、摩托罗拉、京瓷、LG、三星以及索尼爱立信都有定位于这个市场的设备。它们通常不具有触屏支持，内存有限，只内置了一个入门级的相机和音乐播放器。

3. 中高端移动设备

对大众市场来说，想获得良好的互联网体验的主流选择即是这类中高端移动设备。中端设备有良好的用户体验，价格也适中。最近几年来，这种类型的设备被称为社交终端，也就是说，用户可以通过这类设备接入社交网站，如 Facebook 或 Twitter。

这个类别的设备通常有一个中等尺寸的屏幕，有基本的 HTML 浏览器支持，有些支持 3G，带有一个不错的相机、一个音乐播放器，能玩游戏，有些支持触屏操作，可以安装应用。高端移动设备和智能手机间的一个很大的区别是：高端移动设备往往不采用固定费率计划[2]进行售卖，用户必须自己去寻找这样的套餐。从 2011 年开

译注 2：类似于包月包流量的套餐计划。

始，这些设备中的大多数开始支持 WLAN（WiFi），如图 1-3。



图 1-3：诺基亚 X3-02 触屏键盘版：一个中端触屏设备，带有数字键盘和 WiFi

4. 智能手机

市面上有很多智能手机，包括 iPhone、基于 Android 的设备、webOS、Symbian、黑莓（BlackBerry）以及 Windows Phone 手机（见图 1-4）。这是最难被定义的一类移动设备。人们会问，为什么某些中高端设备没有足够“智能”到可以归入智能手机类别？关于智能的定义每年都在变化，事实上，当前市场上最简单的移动设备在 10 年前看来也是非常智能的。



图 1-4：智能手机设备示例

通常当你购买智能手机时，需要签署一份一年或两年期的合约，约定固定的费率和数据流量计划。符合当前关于智能手机的定义的设备，通常会有一个支持多任务的操作系统、一个现代的支持 HTML5 的浏览器，支持无线局域网（WLAN，也称 WiFi）和 3G 接入，还会有一个音乐播放器，以及下列功能：

- GPS（全球定位系统）或 A-GPS（辅助全球定位系统）；
- 数字指南针；
- 可录像的相机；
- TV 输出；
- 蓝牙；
- 触屏；
- 3D 视频加速；
- 加速传感器。



一些多媒体设备也被我们这些 Web 创作者列为智能手机，但它们其实没有电话功能。这些设备包括 Apple 的 iPod Touch 和 Sony 的 PSP。它们和平板电脑唯一的区别是其屏幕尺寸通常小于 3 英寸。

5. 平板电脑

平板电脑通常带有一块大屏幕（6 到 11 英寸之间）、一个完整支持 HTML5 的浏览器，支持 WLAN 接入（WiFi），有的带 3G，支持触屏操作，还具备那些能在智能手机上找到的所有其他功能。

在这个分类中，我们可以找到很多设备，包括：

- 苹果 iPad；
- 三星 Galaxy Tab；
- 黑莓 BlackBerry PlayBook；
- 巴诺（Barnes and Noble）Nook Color；
- 摩托罗拉 Xoom；
- LG Optimus Pad；
- 亚马逊 Kindle Fire；
- 索尼 S1 和 S2。

1.3.2 操作系统和浏览器

本书并未打算深入研究移动设备的生态系统。在《移动网络程序设计》一书中，可以找到一个详细列举了各种操作系统、平台和浏览器的列表。不过，如果打算创建移

动 Web 体验，至少要对操作系统和浏览器有所了解。

在移动设备的世界里，我们将操作系统分为两大类：可识别的操作系统和专有操作系统。移动电话、低中端移动设备上的操作系统大多属于后一类。

对那些安装了可识别操作系统的设备，我们更关心它上面所运行的操作系统，而不是它的设备品牌或产品型号。我的意思是说，我们并不是要为三星的 Galaxy 开发 Web 应用，而是要为 Android 设备开发 Web 应用。iPhone 是一个例外，因为它的平台是专有的，在写作本书的时候，这个平台只支持一个设备：iPhone。（不同版本的设备也遵循相同的规则，对 Web 开发者而言，iPhone 4 和 iPhone 3GS 并没有太大的区别。）

表 1-1 列举了当前市场上所有智能手机和平板电脑所使用的操作系统。

表1-1：智能手机、社交设备以及平板电脑上的操作系统和浏览器

操作系统	厂　　商	内置浏览器	其他浏览器
iOS	Apple	Safari	Opera Mini 和伪浏览器
Android	Google	Android 浏览器	Firefox, Opera Mini, Amazon Silk, Opera Mobile
Symbian	Nokia	Symbian 浏览器	Opera Mini, Opera Mobile
webOS/Open webOS	HP（曾经的 Palm）	webOS 浏览器	
Windows Phone	微软	IE	
Windows Mobile	微软	IE	Opera Mobile
MeeGo	Nokia	Micro 浏览器 /Nokia 浏览器	Firefox
BlackBerry OS	RIM	BlackBerry 浏览器	Opera Mini
Tablet OS	RIM	Tablet OS 浏览器	
S40	诺基亚	诺基亚浏览器	
Bada	三星	三星浏览器	

每个操作系统都有多个版本，有些系统允许用户升级到新版本。每个操作系统都内置一个浏览器，不过用户可以自行安装使用其他浏览器。有时设备制造商或者运营商（你从他们那买的设备）会安装别的浏览器，比如 Opera Mobile 浏览器，或者用它替换掉原有的浏览器。

如果我们将浏览器的调查延伸至中低端设备，还会进一步发现 20 多种其他浏览器，如 Ovi 浏览器、NetFront 浏览器以及 LG 的 Phantom 浏览器。但它们目前都不是 jQuery Mobile 的应用目标。

> **什么是伪浏览器？**
>
> 伪浏览器是一个用户可安装到设备上的本地应用，它们和默认浏览器使用同一个引擎，但是提供了更多的功能。在iOS上有大量这样的例子，如SkyFire或Perfect Browser，它们都使用Safari作为最终渲染引擎，因此，对jQuery Mobile而言，它们是同一个浏览器。

在《移动网络程序设计》一书中有 20 页关于每个浏览器类型及特性的详细介绍。

1.3.3　jQuery Mobile兼容性

jQuery Mobile 框架专注于智能手机、平板电脑以及多媒体设备等触屏设备，它的兼容列表会随着时间的推进以及框架本身的继续演化而改变，因此，这儿没法给出完整的清单。

jQuery Mobile 1.0 版与下列系统的默认浏览器兼容。

- iOS
  iOS 3.2 版开始的 iPhone、iPod Touch 以及 iPad 版 Safari。
- Android OS
  手机及平板版 Android 浏览器。
- BlackBerry OS
  BlackBerry 浏览器，智能手机版 5.0 版及最新版，以及平板电脑版。
- Symbian
  支持触屏设备的 Nokia 浏览器。
- webOS
  自 webOS 1.4 版开始的 webOS 浏览器。
- Bada
  Bada 浏览器。
- MeeGo
  Micro 浏览器及 Nokia 浏览器（内置在 Nokia N9 中）。

- Windows Phone
  Windows Phone/Mobile 6.5 及 Windows Phone 7.0 中的 IE。

- Kindle
  Kindle 3 中的浏览器。

jQuery Mobile 也与下列第三方浏览器兼容：

- Opera Mini，自 5.0 版开始在大多数设备上都能完整支持；
- Opera 移动版，自 10.0 版开始在大多数设备上都能完整支持；
- Firefox 移动版。

这个兼容清单只是让你有一个初步了解，完整的兼容性展示远比这个清单复杂，因为多个操作系统版本与多个浏览器版本相互交织，不同的组合可能产生不同的结果。甚至，这里没有列出的新设备也可能兼容 jQuery Mobile，只要它支持 jQuery Mobile 框架所必需的最少特性。

简单来说，jQuery Mobile 能兼容每个浏览器并提供框架支持的体验。所有现代浏览器都应该在这个清单中。



许多现代浏览器使用基于 WebKit 的引擎，如桌面版浏览器 Safari 或 Chrome。所有现代的基于 WebKit 的移动浏览器都应该能完整兼容 jQuery Mobile。同时，桌面版的 Chrome、Firefox、Safari、Opera 以及 IE 等浏览器都与 jQuery Mobile 兼容。

移动分级浏览器支持

jQuery Mobile 用一个图表来表示它与各个设备的兼容情况（见图 1-5）。如果我是你，我就不会去趟这道分类的混水，不过，要是你真想了解更多信息，可以在 http://jquerymobile.com/gbs/ 查看。



许多现代桌面浏览器，如 Firefox、Google Chrome、Safari 或 IE 也都兼容 jQuery Mobile。虽然 jQuery Mobile 无意于桌面应用，但这个特性在测试时会很有用。另外，后面我们也会看到安装一个模拟环境也很有用。

我相信兼容性问题远比这个表上列出的要复杂，并且一般 Web 设计及开发人员不必太关心。判断一个移动浏览器是否支持某个特性，有比将它们逐一分类更好的办法，其中一个方案就在你手中：使用 jQuery Mobile。

MOBILE GRADED BROWSER SUPPORT

Platform	Version	Native	Opera Mobile 8.5	Opera Mobile 8.65	Opera Mobile 9.5	Opera Mobile 10.0	Opera Mini 4.0	Opera Mini 5.0	Fennec 1.0	Fennec 1.1	Ozone 0.9	Netfront 4.0	Phonegap 0.9
iOS	v2.2.1	A											A
	v3.1.3, v3.2	A						A					A
	v4.0	A						A					A
Symbian S60	v3.1, v3.2	C	C	C		B	C	B			C	C	
	v5.0	A	C	C		A	C	A					A
Symbian UIQ	v3.0, v3.1			C							C		
	v3.2				C						C		
Symbian Platform	3.0	A											
BlackBerry OS	v4.5	C					C	C					
	v4.6, v4.7	C					C	B					C
	v5.0	A					C	A					A
	v6.0	A						A					A
Android	v1.5, v1.6	A											A
	v2.1	A											A
	v2.2	A				*A*		*C*		*A*			A
Windows Mobile	v6.1	C	C	C	C	B	C	B				C	
	v6.5.1	C	C	C	C	A	C	A					

图 1-5：jQuery Mobile 在它的网站上维护着一个浏览器兼容清单

GBS（分级浏览器支持，Graded Browser Support）将移动浏览器分为三个级别：A 级、B 级以及 C 级。对 jQuery Mobile 来说，各级含义如下：

- A 级
  支持 CSS3 媒体查询（media queries）的浏览器。这类浏览器会被 jQuery 团队完整地测试。不过，如果设备不支持，则一些特性会被自动禁用。jQuery Mobile 将提供一个完整的基于 AJAX 动画的体验。

- B 级
  具有增强体验，但是没有 AJAX 导航特性的浏览器。

- C 级
  不兼容 jQuery Mobile 的浏览器。这类浏览器不能从 jQuery Mobile 框架接收任何 CSS 或 JavaScript 代码，因此用户将只能看到一个无样式的 HTML 文件内容。本书后面会提到如何处理这种情况。



PhoneGap 以及原生开发

上面的 jQuery Mobile 移动分级浏览器支持表中，PhoneGap 被当作了一个浏览器。不过，PhoneGap 并不是浏览器，它是一个用于创建混合解决方案（嵌入了 Web 应用的原生应用）的框架。在 iOS、Symbian、BlackBerry、Android 以及 webOS 等平台上，PhoneGap 都被 jQuery 官方支持。

好消息是，你可以使用任何你喜欢的混合解决方案框架，只要设备支持 PhoneGap，jQuery Mobile 就也能工作，这是因为 PhoneGap 不是浏览器本身，而是一个使用了原生浏览器引擎的框架。

简单来说，jQuery Mobile 兼容使用 HTML 创建原生应用。

1.4 HTML5和CSS3

我知道大多数 Web 设计及开发人员都对 HTML5 和 CSS3 感到恐慌。我首先想说：不要担心，jQuery Mobile 将为你处理一切。因此，即使不懂 HTML5 或 CSS3 也可以使用 jQuery Mobile。尽管如此，我仍然鼓励你学习一下 HTML5 和 CSS3，这样，在将来的讨论中，你将对这些新标准有一个更深的认识。

本书不会教读者 HTML5 或 CSS3，不过，了解一些相关知识很重要。很多智能手机、平板电脑内置的移动浏览器都支持 HTML5、CSS3 以及其他 API。

关于 HTML5 有很多可谈论的东西，包括它的历史以及它为移动世界带来了什么。

严格来说，HTML5 是一个发展中的标准，它包含了若干对 HTML 标记语言的改变以及大量 JavaScript 中的新 API（是的，HTML5 的大部分内容是关于 JavaScript API 的）。在非正式的场合中，HTML5 是许多浏览器中的现代特性的汇总，包括 W3C 的正式的 HTML5 标准、其他 W3C API、CSS3 以及非标准扩展。可以在 http://mobilehtml5.org 看到 HTML5 在各移动浏览器中的兼容信息。

jQuery Mobile 使用了大量的 HTML5 特性以便在移动浏览器上提供更好及更快的体验。不过这并不意味着浏览器必须完整地支持 HTML5。事实上，许多早于 HTML5 出现的老浏览器也支持一些 HTML5 标记。在处理动画、渐变、特效以及 UI 渲染时，jQuery Mobile 将尽可能地使用 CSS3。

为了进一步引起你的兴趣，必须得告诉你：使用 HTML5、CSS3 以及一些其他现代技术，可以获得下列特性（无论是否带有 jQuery Mobile 体验）：

- 离线访问；
- 离线存储；

- Websockets；
- 地理位置访问；
- 加速计及陀螺仪支持；
- 动画；
- 2D 及 3D 变换；
- 渐变及视觉特效；
- 视口（viewport）管理（用于浏览器内置的缩放支持）；
- Web 应用安装元数据；
- 与原生应用整合；
- 多媒体支持；
- 绘图（矢量图及位图）；
- 支持自定义字体。

我的博客 http://www.mobilexweb.com/ 上有一些关于这些特性的例子及链接。

1.5 主要特性

作为一个现代框架，jQuery Mobile 项目始于 2010 年 8 月，它包含很多适用于多平台开发的模式及最佳实践。这个框架的主要特性有：

- 跨平台、跨设备、跨浏览器；
- 为触屏设备优化过的 UI；
- 设计为可修改主题及自定义；
- 只使用无侵入性的 HTML5 代码，无需了解任何 JavaScript、CSS 或 API 知识；
- 自动调用 AJAX 来加载动态内容；
- 构建于知名及有良好支持的 jQuery 核心之上；
- 轻量级尺寸，压缩后为 12KB；
- 渐进增强；
- 可访问性支持。

我们已经讨论过其中一些特性，现在让我们深入分析一下其他方面。

1.5.1 使用非侵入性语义的 HTML5

我知道你急切地想看代码，现在代码来了。jQuery Mobile 使用标准及无侵入性的 HTML5 创建 Web 应用，特别适合搜索引擎优化（SEO）以及网站性能优化（WPO）。

```
<!DOCTYPE html>
 <html>
 <head>
        <meta charset="utf-8" />
        <title>My first jQuery Mobile code</title>
        <link rel="stylesheet" href="http://code.jquery.com/mobile/1.0/
            jquery.mobile-1.0.min.css" />
        <script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
        <script type="text/javascript" src="http://code.jquery.com/mobile/
            1.0/jquery.mobile-1.0.min.js"></script>
        <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

 <body>
 <div data-role="page" data-theme="a">
        <div data-role="header">
                <h1>jQuery Mobile</h1>
        </div>
        <div data-role="content">

                <ul data-role="listview" data-inset="true" data-divider-
                    theme="b">
                     <li data-role="list-divider">Summary</li>
                     <li><a href="ch1.html">The Platform</a></li>
                     <li><a href="cap2.html">The Page</a></li>
                     <li><a href="cap3.html">Lists</a></li>
                     <li><a href="cap4.html">Components</a></li>
                </ul>

                <ul data-role="listview" data-inset="true" data-divider-
                    theme="d">
                    <li data-role="list-divider">Links</li>
                    <li><a href="http://www.mobilexweb.com">Mobile Web Blog
                        </a></li>
                    <li><a href="http://www.oreilly.com">O'Reilly Media</a>
                        </li>
                </ul>

       </div>
    <div data-role="footer">
       <h4>&copy; 2011 Maximiliano Firtman @firt</h4>
    </div>
</div>
</body>
</html>
```

从图 1-6 可以看到这个示例在几个移动浏览器上渲染的情况，图 1-7 则显示了不兼容 jQuery Mobile 的情况。如你所见，代码中没有用于初始化或其他用途的 JavaScript 代码，只不过包含了几个 JavaScript 文件。

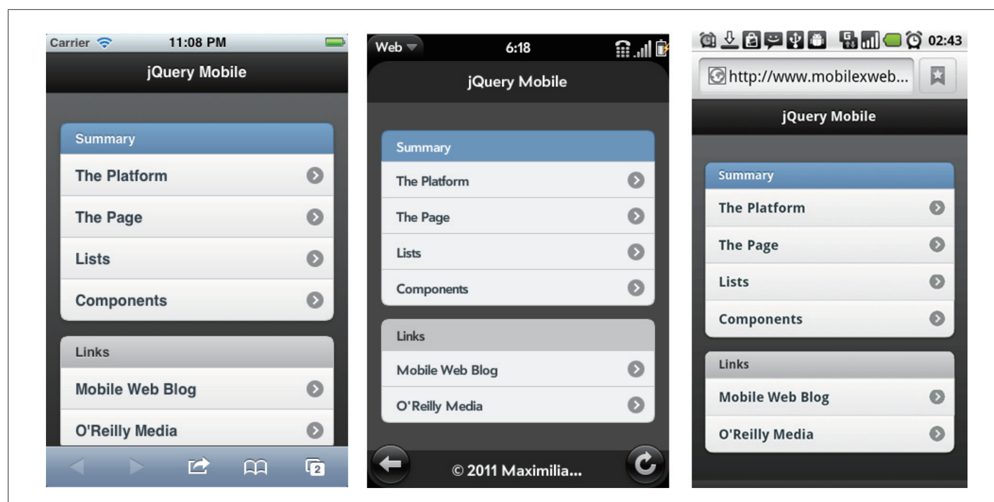


图 1-6：上面的 jQuery Mobile 代码示例在 iOS、webOS 以及 Android 设备上的效果



图 1-7：在不兼容的浏览器上，jQuery Mobile 将降级为一个简单的、功能齐全的 HTML 文件

不要急，下一章我们就开始分析上面的 jQuery Mobile 代码。

1.5.2　渐进增强

渐进增强是一种用于 Web 设计的简单但非常强大的技术，它定义了几个层次的兼容性，允许所有用户都能访问网站的基本的内容、服务以及功能，同时在那些对标准支持更好的浏览器上提供增强的体验。jQuery Mobile 完全使用这个技术构建。

渐进增强这个术语由 Steven Champeon 在 2003 年提出（http://www.hesketh.com），虽然这个技术并不是专门为移动互联网定义的，但它却特别适合移动互联网设计。

渐进增强有以下核心原则：

- 在所有浏览器上都能访问基本内容；
- 在所有浏览器上都能使用基本功能；
- 语义标签包含了所有内容；
- 增强布局由外部链接的 CSS 提供；
- 增强行为由不冲突的、外部链接的 JavaScript 提供；
- 尊重终端用户浏览器的偏好设置。

这个列表看起来是不是像 jQuery Mobile 的特性列表？是的，就是这样。jQuery Mobile 应用也能在不支持 CSS 或 JavaScript 的入门级的浏览器上工作，这对移动互联网应用来说是一个很好的特性。

1.5.3　可访问性支持

维基百科上的定义：

> Web 可访问性指的是让网站对所有正常人及残疾人都可用的一系列做法。在正确地设计、开发以及编辑的网站上，所有用户都可以平等地访问网站提供的各种信息及功能。

移动浏览器上的 Web 可访问性浪潮才刚刚兴起。不过，jQuery Mobile 已经完全兼容 W3C 关于浏览器兼容的 WAI-ARIA 标准（http://www.w3.org/TR/wai-aria/）。在写作本书的时候，还只有带 VoiceOver 特性的 iOS 4.0 或更新版本才兼容这个标准。

因此，在 iPhone、iPod 以及 iPad 上，基于 jQuery Mobile 的 Web 应用都能为有视力障碍的人士提供可访问体验。

1.6　测试Web应用

我们提到过，基于 jQuery Mobile 的 Web 应用可以在绝大多数现代桌面浏览器上运行，不过，最好还是能在更精准的环境中进行测试（见图 1-8）。

要在不同的环境中测试移动 Web 应用，可以使用：

- 真实的设备；
- 远程实验室；
- 仿真器；
- 模拟器；
- 大量的朋友[3]。

译注 3：指让朋友们用自己的手机帮忙测试。

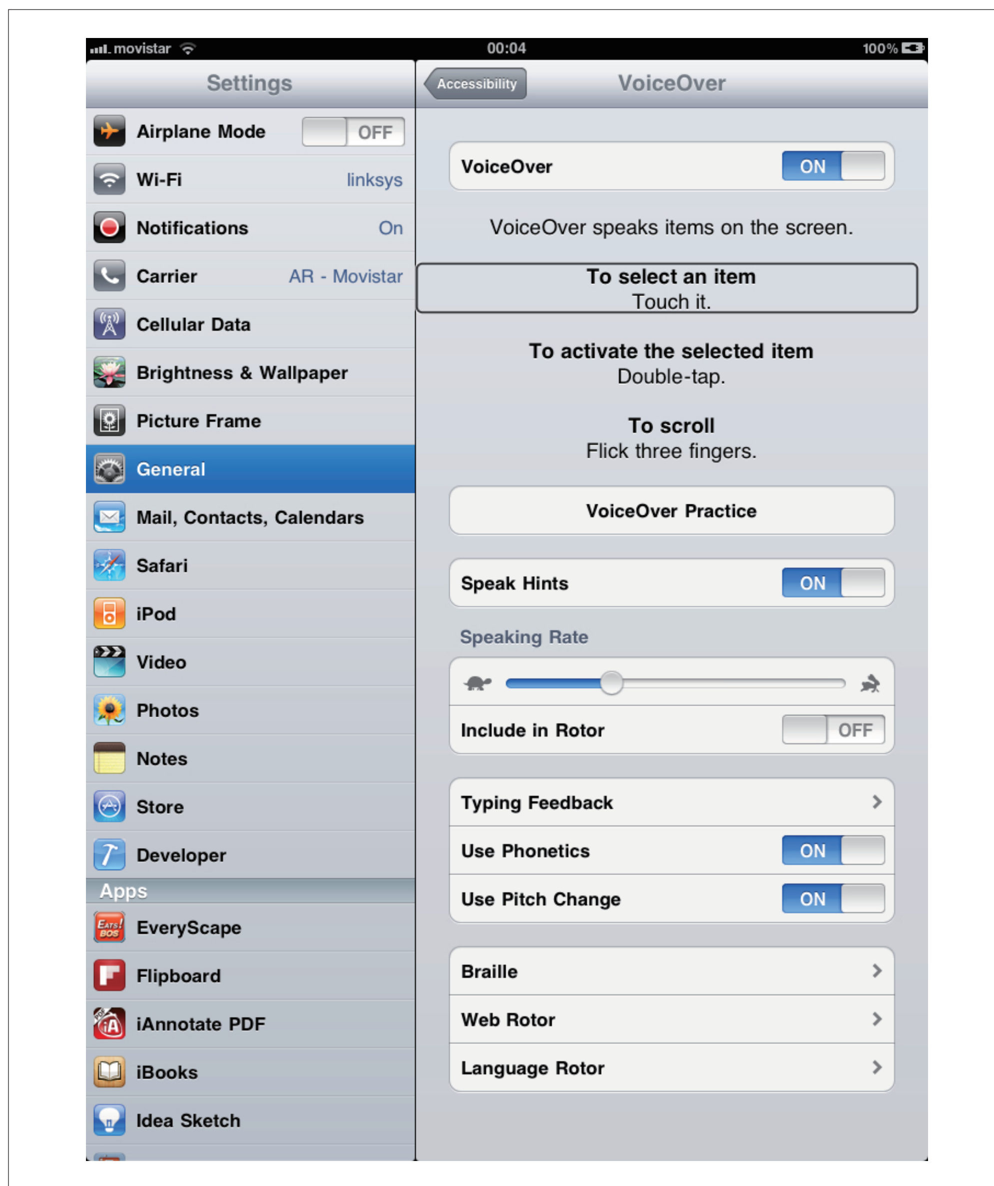


图 1-8：在 iPad 上测试 jQuery Mobile 的可访问性

1.6.1 仿真器与模拟器

仿真器与模拟器是最有用的工具。总的来说，仿真器是一个软件，将已编译的代码从原始架构翻译为当前运行平台的代码，让我们能在一个操作系统上运行另一个操作系统及程序。在移动开发的世界里，仿真器是仿真移动设备的硬件及操作系统的

桌面程序，可用于测试及调试应用，以及查看应用的工作状况。浏览器甚至操作系统并不会意识到它们正运行在仿真器上，所以相同的代码也能在真实的设备上运行。

我们的移动开发环境中也应该添加一些项目及配置管理的经典工具，如 bug 追踪、版本管理以及项目管理等工具。图 1-8 显示了如何在 iOS 4.0 或更新版本的 iPhone、iPod 或 iPad 上测试 jQuery Mobile 的可访问性。打开菜单“设置➤通用➤可访问性”并激活 VoiceOver，然后闭上眼睛，使用手指及耳朵浏览你的网站。

图 1-9 展示了一个 Android 仿真器在桌面上提供完整的 Android 系统图形化的效果，可用于仿真 Galaxy Tab 或 Nook Color 等平板设备。



图 1-9：一个 Android 仿真器

仿真器由厂商开发，向开发者免费提供，有些是独立程序，有些则与开发原生应用的软件开发工具包（SDK）捆绑在一起。

也有一些操作系统仿真器，它们与任何真实的设备硬件无关，只仿真了操作系统。Windows Mobile 和 Android 都有这类仿真器。

另一方面，模拟器则是相对简单的程序，它只模拟设备的一些行为，但不能模拟

硬件，也不是基于真实的操作系统设计的。这些工具比较简单，比仿真器用途少。模拟器可能由设备制造商创建，也可能由其他为开发者提供模拟环境的公司提供。在移动浏览方面有一些像素级的模拟器，还有一些则既没有替换桌面浏览器（如 Firefox 或 Safari）的外观，也不模拟渲染引擎。图 1-10 显示了 Mac 中免费 iOS 模拟器提供的 iPad 模拟。其他平板电脑，包括 Windows 或 Linux 桌面机，也有类似的模拟器。



图 1-10：iOS 模拟器

就算使用仿真器，最终渲染效果以及性能也可能和真实设备上不完全一样。因此，使用真实设备来测试是一个不错的做法，即使只在一些重要设备上测试。

对手机网页开发来说，可以找到 Nokia、Symbian、BlackBerry、Android、webOS 以及 Windows Mobile 的仿真器，还有来自苹果公司的用于 iPhone 和 iPad 的模拟器（不过只能在 Mac OS X 上使用）。

还有一些基于浏览器的仿真器（最终能在许多不同的平台上运行），如 Opera Mobile 仿真器。

表 1-2 列出了可下载的仿真器与模拟器。

表1-2：可下载的仿真器与模拟器

名　　字	平　　台	类型	可用浏览器	Windows	Mac	Linux
iOS 模拟器	iOS	模拟器	Safari	否	是	否
Android 仿真器	Android	仿真器	Android 浏览器，可下载	是	是	是
HP webOS 仿真器	webOS	仿真器	webOS 浏览器	是	是	是
Nokia Symbian 仿真器	Symbian	仿真器	内置浏览器，可下载	是	否	否
Windows Phone 仿真器	Windows Phone	仿真器	IE	是	否	否
Nokia S40 仿真器	Nokia OS	仿真器	S40、Ovi 浏览器、Opera Mini	是	否	否
BlackBerry 模拟器	BlackBerry OS	仿真器	BB 浏览器，可下载	是	否	否
BlackBerry PlayBook 模拟器	Tablet OS	仿真器	Internal 浏览器	是	是	是
Opera 移动仿真器	很多	浏览器仿真器	Opera Mobile	是	是	是
Opera Mini 模拟器	很多	在线浏览器仿真器	Opera Mini	是	是	是
PhoneGap 模拟器	很多	模拟器	PhoneGap 混合	是	是	是
Adobe 设备中心	很多	模拟器	很多	是	是	否

最新的仿真器下载链接列表见 http://www.mobilexweb.com/emulators。

1.6.2　远程实验室

远程实验室是一种 Web 服务，我们可以远程使用不在同一个物理位置的真实的设备。这是一个简单但强大的解决方案，只需鼠标一点，即可访问世界各地数以千计的连接到真实网络的真实设备。可以把它想象为移动电话的远程桌面。

这个市场上最有用的服务如下：

- Keynote DeviceAnywhere（商业）
  http://www.deviceanywhere.com
- Perfecto Mobile（商业）
  http://www.perfectomobile.com
- Nokia Remote Device Access for Symbian and MeeGo（免费）
  http://www.mobilexweb.com/go/rda
- Samsung Lab.Dev for Android（免费）
  http://www.mobilexweb.com/go/labdev

关于这个主题的最新信息见 http://www.mobilexweb.com/go/labs。

第 2 章

框架起步

2.1 准备文档

让我们挽起袖子，创建一个典型的 jQuery Mobile Web 应用模板。

2.1.1 需求

我们的 HTML5 文档需要包含：

- jQuery 核心 JavaScript 文件；
- jQuery Mobile 核心 JavaScript 文件；
- jQuery Mobile 核心 CSS 文件；
- jQuery Mobile 主题 CSS 文件（可选）。

另外，对于某些 UI，jQuery Mobile 会使用一系列 PNG 文件，不过不用显式包含它们。还有一个版本的 CSS 文件同时包含了核心文件和默认主题。

在编码之前，首先要决定的是资源文件的托管。有两个选择：

- 将所有文件托管在项目中；
- 使用 CDN（内容分发网络）；

2.1.2 托管文件

如果想将所有文件和 Web 应用托管在一起，需要从 http://jquerymobile.com/

download 下载最新的 ZIP 包。该 ZIP 包的名字中包含框架的版本，例如 jQuery.mobile-1.0.zip。

jQuery Moile 包中不含 jQuery 核心，后者需要从 http://jquery.com 下载（推荐下载生产版本）。

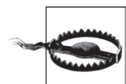
jquery.mobile-XX.zip 包的结构如下。

- 示例文件夹；
- 图片文件夹；
- jquery.mobile-XX.css；
- jquery.mobile-XX.js；
- jquery.mobile-XX.min.css；
- jquery.mobile-XX.min.js；
- jquery.mobile.structure-XX.css；
- jquery.mobile.structure-XX.min.css。

其中，XX 是版本号，包括发布类型，如：1.1b1 代表 1.1 Beta1 版，1.0rc2 代表 1.0 候选版本 2，或 1.0 代表 1.0 最终版。

程序包里有两类 JavaScript/CSS 文件，一类文件名中有 min 后缀，另一类没有。

带 min 后缀的文件推荐用于生产版本，因为它们已被压缩（去掉了空格、注释以及换行）。如果需要在 jQuery Mobile 中调试，则可使用不带后缀的版本。



jQuery Mobile 1.0 需要 1.6.4 版的 jQuery。不要使用更新的 jQuery，因为可能有兼容问题。如果正在使用新版本的 jQuery Mobile，检查一下它的文档，确定它需要哪个版本的 jQuery。

一般情况下，需将下列文件添加到项目的根目录下：

- jquery-XX.js（来自 jQuery 核心）；
- 图片文件夹；
- jquery.mobile-XX_min.js；
- jquery.mobile-XX_min.css。

如果使用 PhoneGap 或其他离线 / 混合机制创建 Web 应用，那么最好将这些文件嵌入到程序包中，这样 Web 应用就可以离线工作了。

在本书后面的章节中可以看到，那些带结构名的文件在创建自定义主题时很有用。

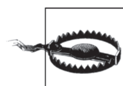
> **jQuery Mobile的许可**
>
> jQuery Mobile（以及作为核心的jQuery）是免费的，它们使用MIT或GPL第二版双协议开源。对大多数项目来说，推荐使用MIT协议，无须从我们这儿获得授权。唯一需要注意的是不要删除或修改文件顶部的版权信息。有任何疑问，请参考http://jquery.org/license。

在自己的服务器上托管这些文件时，需要确认服务器端会在客户端支持时启用 gzip 压缩，这将令 jQuery Mobile 的 JavaScript 和 CSS 传输及加载时间减少 80%。如果不知道怎么做，请咨询你的服务器提供商，或者查看 http://mobilexweb.com/go/performance。

2.1.3　使用CDN

有一个更简单的使用 jQuery Mobile 的方法：使用 CDN（Content Delivery Networks，内容分发网络）。jQuery CDN 是为我们在互联网上托管相关文件的公共服务器。这个方法有优点也有缺点。

最大的缺点是我们的 Web 应用将只能在公共 CDN 在线时才能工作。当然，它们正是为此而建的，会竭尽所能地保证每周 24 小时不间断服务。不过，有些项目不能依赖第三方服务，如离线 Web 应用。



使用 jQuery Mobile 创建混合或原生应用（如 PhoneGap 应用）时，不应该使用 CDN。如果设备掉线了，甚至连用框架显示一个友好的警告都做不到。创建 HTML5 离线 Web 应用时，如果已将 CDN 文件加入了应用程序清单（manifest），则可以使用 CDN。

使用 CDN 的主要优点如下。

- 不用下载，马上就可使用 jQuery Mobile。
- 你的服务器分发的文件更少，从而节约带宽。
- Web 应用将从缓存中受益：如果用户曾访问过另外的使用同一个 CDN 上的 jQuery MobileWeb 应用，那么他的浏览器缓存可能已经有所有相关资源了。
- 对大多数共享主机来说，jQuery Mobile 资源从 CDN 下载会更快。
- Web 应用的性能将从不同的域名中受益。
- 对有些 CDN 来说，链接到一个地址时总是返回最新的版本。不过，有时我们并不推荐这种特性，因为如果没测试，你永远不会知道你的 Web 应用在新版本的框架下会有什么表现。

本书中没有关于移动浏览器 WPO（Web Performance Optimization，网站性能优化）的内容。如果想了解更多相关信息，请访问 http://www.mobilexweb.com/go/performance 并订阅 Steve Souders 的博客 http://stevesouders.com。

对 jQuery 核心来说，有一些可选择的 CDN：

- 官方 jQuery CDN；
- 微软 jQuery CDN（http://www.asp.net/ajaxlibrary/CDN.ashx）；
- Google AJAX 库接口（http://code.google.com/apis/libraries/）。

写作本书的时候，官方 jQuery CDN 和微软 CDN 都托管了 jQuery Mobile 的文件。

使用 CDN 真的很简单，只需复制粘贴对应的 JavaScript 或 CSS 外部文件的 URL 就完成了。

在 http://jquerymobile.com/download/ 可以看到一个可直接复制粘贴使用的代码片断，看起来像这样：

```
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.0/jquery.
    mobile-1.0.min.css" />
<script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
<script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js">
    </script>
```

如果使用自己的主题（后面的章节中有介绍），则需要使用下面的代码：

```
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.0/jquery.
    mobile.structure-1.0.min.css" />
<!-- 这儿放置我们的主题 CSS 文件 -->

<script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
<script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js">
    </script>
```

如上所示，只需引入三个托管在 code.jquery.com 的压缩版外部资源即可，不需要下载图片、CSS 或 JavaScript 代码等东西即可开始使用。记得检查下网站上可用的最新版本。

最新构建版本

如果希望总是使用最新版的 jQuery Mobile，jQuery CDN 也提供了对应资源，可直接嵌入代码。记住，这些版本会自动更新，所以尽管你的代码现在能正常工作，但未来却可能出现问题。这个选择一般用于开发或测试，不要忘了，它可能包含不稳定的开发代码。

使用最新构建版本的代码如下：

```
<link href="http://code.jquery.com/mobile/latest/jquery.mobile.min.css"
    rel="stylesheet" type="text/css" />
<script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
<script src="http://code.jquery.com/mobile/latest/jquery.mobile.min.js">
    </script>
```

使用最新开发构建版本的好处是可以使用那些在最新的生产版本中还没有的新特性。

2.1.4 主HTML5模板

要创建 jQuery Mobile Web 应用，只需创建一个空的 HTML5 文件。如果之前没写过 HTML5 文档的话也不要担心，这很容易，和写 HTML 4.01 文件很像。

HTML5 文件的 DOCTYPE（文档的第一行）非常简单：`<!DOCTYPE html>`。

另一个改变是 `head` 标签中的 `meta charset` 标签：`<meta charset="utf-8" />`。

限于篇幅，本书无法深入讨论自 HTML 4.01 的改变，不过就目前来说，这些改变在一些不支持 HTML5 但是能运行 jQuery Mobile 的老设备上也是兼容的。



如果使用 CS5 及更新版本的 Dreamweaver，可以从新文档窗口的文档类型下拉列表中选择 HTML5 来创建一个 HTML5 空模板。如果你的 Dreamweaver 没有这个选项，则需要先通过 Adobe.com 网站或 Adobe 升级程序下载 11.0.3 或之后的更新。

jQuery Mobile 开发团队官方推荐将相关的 JavaScript 及 CSS 资源包含在 `head` 标签中（自己托管或基于 CDN），再添加一个视口（viewport）`meta` 标签：

```
<!DOCTYPE html>
 <html>

 <head>
        <meta charset="utf-8" />
        <title>Your Title</title>
        <link rel="stylesheet"
          href="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css" />
        <script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
        <script type="text/javascript"
         src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js">
             </script>
      <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

 <body>

 </body>

</html>
```

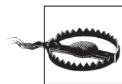
这就可以了！我们刚刚创建了一个空 jQuery Mobile 文档。当然，它的 body 中还没有内容，稍后我们会回来继续完成它。

1. 视口

视口（viewport）是页面所占的区域。可以指定它的宽度和高度，它可以比屏幕的全部可见区域更大或更小，这正是移动浏览器的比例或缩放特性的用武之地。创建一个移动友好的网站时，它应该不需要缩放，或者说它的默认宽度与设备屏幕可见区域的宽度相同，这时可以告诉浏览器使用 1：1（视口区域：可见区域）的比例。让我们看看图 2-1，其中显示了 iOS 上的 jQuery Mobile 如果不指定视口会怎么样。



图 2-1：jQuery Mobile 中，需要定义一个视口 meta 标签，以免加载 Web 应用时出现这种 UI 问题



jQuery Mobile 的 alpha 版本曾经会自动添加视口定义标签。如果你正在迁移一个使用 jQuery Mobile alpha 版的老 Web 应用，记得要在页面中添加视口 `meta` 标签。

适用于 jQuery Mobile 的典型视口 `meta` 标签类似这样：

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

也可以指定禁止用户改变页面比例（使用手势或按钮）：

```
<meta name="viewport" content="width=device-width, initial-scale=1,
    user-scalable=no">
```

2. JavaScript的性能

在继续完成我们的 Web 应用之前，需要讨论一下性能问题。网站性能优化（WPO）领域有一个广为人知的经验，即从性能的角度看，在 `head` 中插入外部脚本标签是不好的。这绝对正确，可以读一下 Steve Souders 的精彩著作《高性能网站建设指南》（*High Performance Web Sites*）。

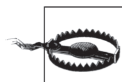
不过，从 jQuery Mobile 的角度来看，把两个脚本（jQuery 和 jQuery Mobile）移到 HTML 文件尾部的做法会带来一个不想要的结果：在框架下载并执行完成之前的一小段时间里，Web 应用将先显示无 CSS 样式的 HTML，就算我们把 CSS 文件放在 `head` 中还是会这样。

这是 jQuery Mobile 框架中使用的渐进增强方法造成的。没有对应的 JavaScript，相关的 CSS 文件对渲染毫无作用。

因此，最好把它们都放在 `head` 中，即使这样会造成一点性能损失也是必要的。

2.2 Adobe Dreamweaver的支持

自 CS5.5 版开始，Adobe Dreamweaver 工具中内置了对 jQuery Mobile 的官方支持。可以从 http://www.adobe.com/go/dreamweaver 下载免费试用。



Dreamweaver CS5.5 的第一个版本包含了 jQuery Mobile alpha 3，而不是最新发布的版本。可以在 http://mobilexweb.com/go/dwjqm 查看如何更新到最新版本的介绍。可以总是以 alpha 3 版本开始，然后手工把它改为最新版本。Dreamweaver 的后续版本中将包含它发布时 jQuery Mobile 的最新稳定版本。

官方支持中包括创建一个使用 jQuery Mobile 的页面。要实现这一点，只需打开 Dreamweaver，选择菜单“文件”➤“新建”，选择“从模板新建”➤ Mobile Starters，接下来就可以从以下三个模板中选择了（见图 2-2）：

- CDN 上的 jQuery Mobile；
- 使用本地文件的 jQuery Mobile；
- 使用本地文件的 jQuery Mobile，包括 PhoneGap 支持（本书后面的章节会涉及）。

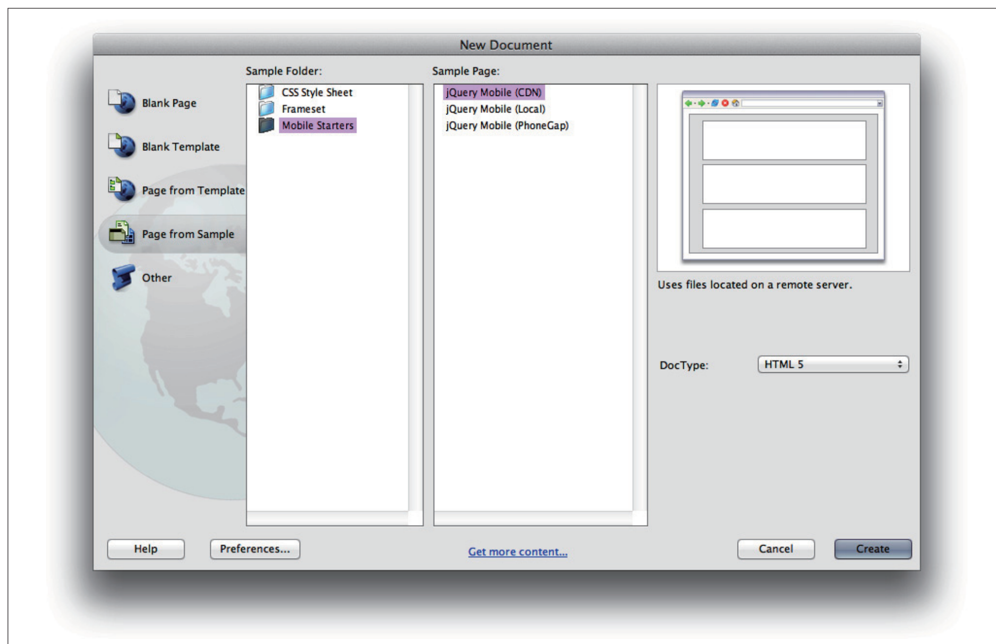


图 2-2：Dreamweaver CS5.5 版开始支持从头创建 jQuery Mobile 模板

每个模板都以一个 jQuery Mobile 文档开始，带有四个相互连接的页面。

使用 Dreamweaver 最大的好处不是模板，而是代码语法助手。键入 `data-` 就可以得到所有可能的 jQuery Mobile `data-*` 值的列表，或者得到所有 `data-*` jQuery Mobile 属性（至少需要 alpha 3 及以后的版本）的列表。



使用 Dreamweaver CS5.5 及更新版本的多屏预览方法，可以看到 jQuery Mobile 是如何适应不同的屏幕尺寸及方向的，包括智能手机和平板电脑。

在“插入”菜单下还有一个叫“jQuery Mobile”的新选项，其中包含本书中讲到的大多数 UI 组件的代码片断。

预览文件

想在 Dreamweaver 中查看 jQuery Mobile 的实际效果，需要使用实况视图（Live View）功能，如图 2-3 所示。

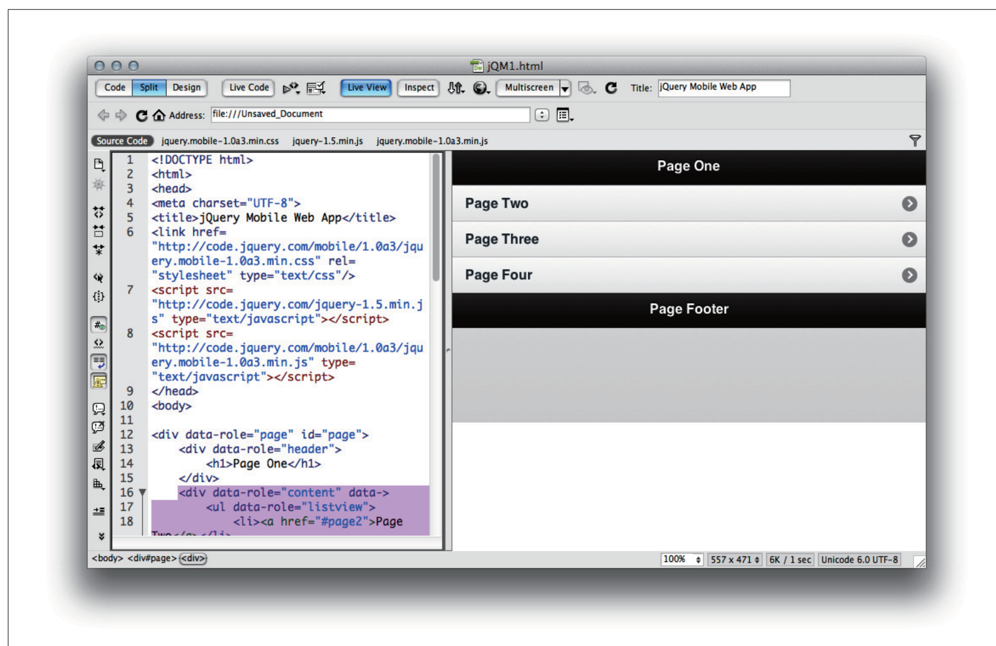


图 2-3：可以使用 Dreamweaver 中的实况视图查看 jQuery Mobile 的实际效果

2.3 架构

jQuery Mobile 使用一种非常简单但极为强大的方法来定义 Web 应用的内容。我们已经讨论过，jQuery Mobile 框架使用的是一种非介入性方法，这意味着我们的 HTML 文档甚至在 jQuery Mobile 没有正常加载时也能工作。

jQuery Mobile 框架的主要单位是页面（page）。这个页面是不是和普通的 HTML 文件一样？不是的。让我解释一下，一个页面仅仅是一个带有指定 `role` 属性的 `div` 元素。一个 HTML 文档能包含一个页面，也能在同一个文件中包含很多个页面。对大多数 Web 设计师来说这是一个新概念。

使用简单的 HTML 标签，比如 `a` 标签，我们可以连接到同一个 HTML 文档中的页面，也能连接到外部 HTML 文档中的页面。

与卡片类似

在移动互联网领域，在一个文档中嵌入多个页面的做法已经有10多年的历史了。现在已经废弃的WML（Wireless Markup Language，无线标记语言）标准允许在同一个文档中插入多个可视页面，目的是减少延迟及下载时间。jQuery Mobile继承了这个思想，不过用的是普通的HTML和JavaScript。

在 WML 世界里，每个页面被称为一个卡片（card），一个 WML 文档则被称为一个面板（deck）。WML 文件使用 `card` 标签来定义文档中的页面，而 jQuery Mobile 通常使用带有特定 `role` 属性的 `div` 标签。

2.3.1 角色

jQuery Mobile 使用标准的 HTML 标记，如 `div` 标签。需要在这个 `div` 上定义一个角色（即 `role` 属性），以便告诉框架如何处理它。jQuery Mobile 中使用属性 `data-role` 来定义 role，例如 `<div data-role="page">`。

jQuery Mobile1.0 中可用的主要的 role 见表 2-1，本书将逐一讲解它们。

自定义`data-*`属性

HTML5中有一个（由W3C标准定义的）名为自定义数据属性的特性，可用来在标签上添加任何形如`data-<something>`或`data-*`的自定义属性，同时保持文档符合HTML规范。这个在不破坏标签的有效性的同时为其添加自定义元数据的特性很有用。

jQuery Mobile经常使用这个特性来定义用于框架的自定义属性。不要困惑，`data-role`不是HTML5的新属性，它只是框架与我们之间的一个默认契约。

自定义属性最大的好处是它们在不支持HTML5的浏览器上也能工作，不会出现严重问题。

如果你使用的是Adobe Dreamweaver CS5.5及之后的版本，在一个HTML元素中键入`data-`时即可看到自动的jQuery Mobile提示。

表2-1：jQuery Mobile 1.0中可用的主要role

role	描　　述
`page`	定义一个页面，这儿的页面是 jQuery Mobile 中用于显示内容的单位
`header`	页面的头部
`content`	页面的内容

（续）

role	描　　述
footer	页面的页脚
navbar	定义一个导航条，一般位于 header 中
button	渲染为一个可视化的按钮
controlgroup	渲染一个组件
collapsible	页面中可折叠的内容面板
collapsible-set	一组可折叠的面板（手风琴布局）
fieldcontain	用于表单域的容器
listview	由多项内容组成的列表
dialog	对话页面
slider	用于布尔值的可视化滑块
nojs	在兼容 jQuery Mobile 的浏览器上会被隐藏的元素

2.3.2　主题

jQuery Mobile 使用一个强大的主题机制来定义用户界面的可视化展现。主题及其自定义会在本书后面的章节讲解，但有一个重要概念现在就应该知道：主题中的每一个 UI 元素（如页面、按钮或组件）都可以使用不同的色卡（swatch）。



直到 1.0 版，jQuery Mobile 还只包含了一个默认主题，不过在 http://jquerymobile.com/themeroller 有一个主题定制器（一个在线 Web 应用，可用于创建自己的主题），通过它无需直接编辑 CSS 文件即可以定义自己的主题。

所谓主题，是指一组对排版、样式以及颜色的定义。每个主题都包含一组色卡，在应用中我们可以随时修改这些色卡。色卡为元素显示提供了不同的选择。色卡由 a 到 z 的字母定义，默认主题包含了从 a 到 e 的定义，为自定义色卡预留了很多字母。

表 2-2 列出了图 2-4 中的颜色对应的色卡约定。

表2-2：色卡约定

字　　母	描　　述	默认主题的颜色
a	最高视觉优先级（一般用于工具条）	黑
b	次高视觉优先级	蓝
c	基本优先级（适用于大多数情况的默认色卡）	银
d	次高优先级的替代方案	灰
e	强调	黄

在 jQuery Mobile HTML 元素上定义色卡使用 `data-theme` 属性，值为一个字母，例如：`data-theme="e"` 表示使用强调色卡。

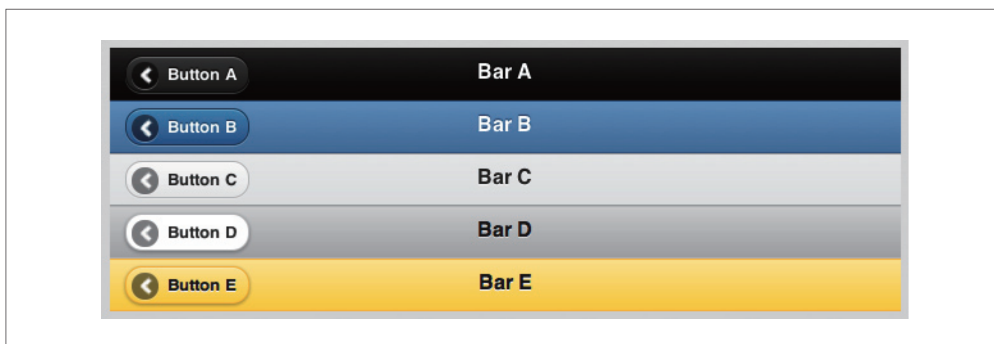


图 2-4：从 a 到 e 的默认主题

一般来说，Web 应用中大多数元素的色卡都可以修改，如页面、列表、按钮、元素、表单元素以及工具条。如果不想同时修改所有同类元素，就可以使用色卡来高亮一个元素而不影响其他元素。

色卡使用层叠机制，就是说，如果一个容器元素定义了一个色卡，除非另有定义，否则它的子元素也将使用这个色卡。

2.3.3 页面

我们知道，*页面*是 jQuery Mobile 的主要单位。一个典型的页面可分为页头、内容、页脚三个部分，其中只有内容部分是必不可少的。各个部分使用带对应 role 的 `div` 标签声明：

```
<div data-role="page">

   <div data-role="header">

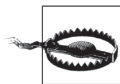
   </div>
   <div data-role="content">

   </div>
   <div data-role="footer">

   </div>

</div>
```

包括页面、页头、页脚以及内容在内的各个部分都可以各自从当前主题中选择一个色卡。



只有一个页面的文档可以不添加 `page-role` 元素，这时框架会自己添加。不过最好还是加上，这会让代码更加清晰，同时将来有修改时也更安全。

图 2-5 显示了一个典型 jQuery Mobile 文档的图解。记住页面需要放在一个引入了 jQuery Mobile 的 HTML5 文档的 `body` 中。

图 2-5：jQuery Mobile 文档中的一个典型页面，其中包含了页头、内容区以及一个可选的页脚

大多数设备上 jQuery Mobile 都可以自行管理方向（纵向或横向），并自动调整 UI 以适应视口。



可以使用值为 nojs 的 role 来提供只在 B 级或 C 级浏览器上显示的内容，如 <div data-role="nojs">。这些内容在 A 级浏览器上会自动隐藏。

在基于 Android、webOS 以及 iOS 等的兼容设备上，jQuery Mobile 会尝试操作初始滚动位置以便隐藏浏览器地址栏，这会让应用的外观和感觉更像原生应用。不过，只有在内容足够高可以填满可见区域时这个方法才管用，否则，地址栏始终会显示。

1. 页头与页脚

在页头和页脚可以插入任何 HTML 内容，但由于标准 jQuery Mobile 样式表的限制，最好在页头中使用 h1，在页脚中使用 h4，以便取得最好的 UI 渲染效果。本书后面的章节会讲到如何自定义这些 UI。

页脚是可选的，不过在 Web 应用的导航 UI 中页头通常是必需的。页头结构已经被预定义并划分为三个子区域：左侧、标题以及右侧。

后面我们会涉及左侧及右侧区域，现在只需要知道它们用于放置操作按钮。

jQuery Mobile 会自动从 hX 标签（如 h1 或 h2 元素）中提取标题。可显示标题的空间是有限的，如果标题太长，它会被自动截断。在大多数兼容设备上，被截断的标题末尾会显示省略号，如图 2-6 所示。页脚也有相同的行为。

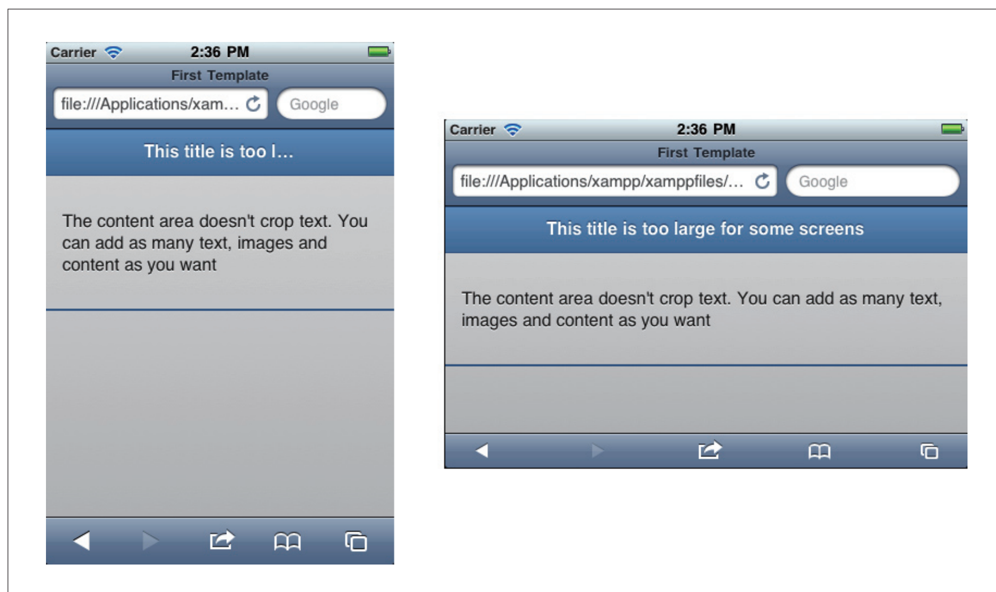


图 2-6：如果标题太长，jQuery Mobile 会在后面加上省略号

2. 内容

内容区域可以包含任意 HTML 代码。一般我们会使用一些框架自带的带样式的控件，如按钮、列表或表单。

图 2-7 显示了改进后的示例：

```
<div data-role="page">

   <div data-role="header">
      <h1>Our first webapp</h1>
   </div>
   <div data-role="content">
      <p>This is the main content of the page</p>
   </div>
   <div data-role="footer">
      <h4>More on mobilexweb.com</h4>
   </div>

</div>
```

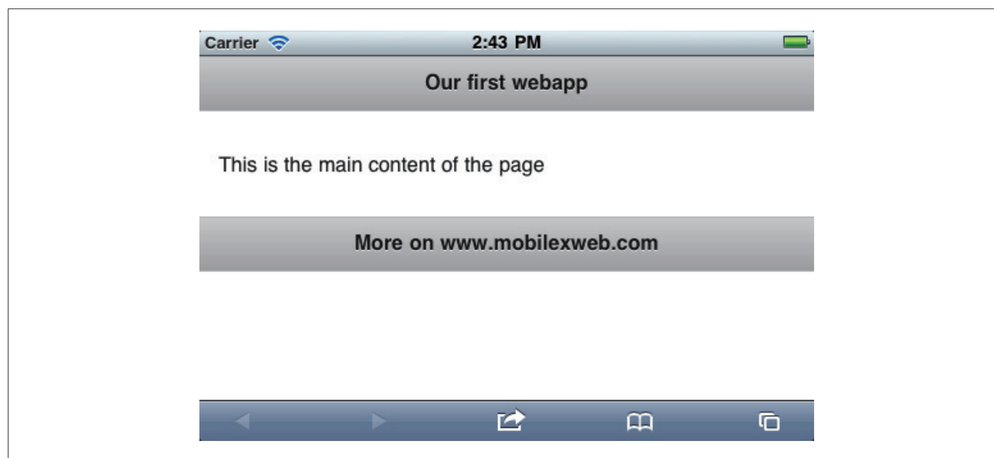
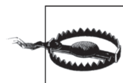


图 2-7：默认情况下，每个 role（页头、内容、页脚）都有一个漂亮的 iOS 风格的外观



jQuery Mobile 样式表无法处理那些在页面之内，但在页头、内容或页脚之外添加的内容，如果不手工处理，这些内容就会成为 UI 错误。

2.4 导航

页面间的导航使用标准 a 元素即可，jQuery Mobile 会神奇地把剩下的事情搞定。

首先，有这么几种超链接类型：

- 链接到同一个文档（这种文档被称为多页面文档）中另一个页面的内部链接；
- 链接到另一个 jQuery Mobile 文档中的页面的外部链接；
- 链接到非 jQuery Mobile 文档的绝对链接；
- 移动应用专有链接。

jQuery Mobile 页面之间的链接（前两种类型）有两种特别的链接行为：

- 在支持的设备上（如 iOS 及基于 Android 的设备），显示页面间的切换动画；
- 在浏览器中（而不是在无边框的本地应用中）运行时，点击浏览器的后退按钮会返回第一个页面。

jQuery Mobile 的一个很棒的特性是它能监听浏览器的后退按钮，并在其被点击时给我们一个明确的后退导航选项。一些设备（如基于 Android 及 BlackBerry 的设备）有硬件后退按钮，它们照样可用于返回前一个页面。



多页面文档中，DOM 中的第一个页面会在第一次加载时显示。

在后退事件触发时从第二个页面转到第一个页面的导航也使用相同的过渡效果，不过顺序相反。

这种后退模式的灵感来自 iOS 及其他移动 OS 的用户界面模式。jQuery Mobile 使用栈来维护用户访问的页面，这样就能随时返回任意访问过的页面。

导航完全由框架完成。jQuery Mobile 会操作当前 URL，在原来的 URL 后面添加散列值。在有些设备上，这个操作会产生一个奇怪的效果：如果地址栏处于隐藏状态，用户导航到另一个页面时，地址栏会出现一会儿，然后再隐藏。这种效果只在基于浏览器的 Web 应用上出现，可以使用本书后面章节中要介绍的全屏的 Web 应用或混合原生应用来避免。

2.4.1 后退按钮

在页面上使用 `data-add-back-btn="true"` 属性将在页头左侧添加一个“后退”按钮。按钮的颜色与色卡可以分别使用页面上的 `data-back-btn-text` 和 `data-back-btn-theme` 来设置。如：

```
<div data-role="page" data-add-back-btn="true" data-back-btn-text="Previous"
     data-back-btn-theme="e">

</div>
```

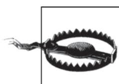


每个浏览器都提供了一个后退按钮，有些是触摸按钮形式，有些是硬件按键形式。因此，创建基于浏览器的体验而不是无边框的 Web 应用（如安装到本地的 Web 应用）时，再为后退创建一个可见的按钮是一种重复劳动，而且会占用宝贵的屏幕空间。不过创建安装到本地的无边框的 Web 应用时，则总是应该包含一个可见的后退按钮。

2.4.2 内部页面链接

前面提到过，一个 jQuery Mobile 文档可以包含多个页面。要实现这一点，可以将这些页面添加为 `body` 的子元素，每个页面都需要使用标准 HTML 方法定义一个 `id`（即 HTML 元素的 id 属性），这样 jQuery Mobile 就可以访问它们了。

在链接的 `href` 属性中使用 `#<id>` 即可创建指向当前文档中的其他页面的链接，这儿的 `<id>` 即目标页面的 `id`，例如：`<a href="#next">`。



外部页面的链接必须与当前页面同域，或者与当前页面在同一个本地应用的包里。对于那些指向不同域的文档的链接，除非在 JavaScript 中开启了跨域 AJAX 加载，否则这些链接将被当作绝对外部链接。

默认情况下，jQuery Mobile 使用文档的 title 元素的内容作为浏览器的标题。标题对基于浏览器的应用很有用，它会显示在某些浏览器的顶部，另外，在用户将当前页面添加为书签时也会用到标题。可以指定页面的可选属性 `data-title`，以便用户访问内部页面时更新标题。

来看一个例子（参见图 2-8）：

```
<body>

<div data-role="page" id="page1">

   <div data-role="header">
      <h1>First page</h1>
   </div>
   <div data-role="content">
      <p>This is the main content of the page.</p>
      <p>You can go to the <a href="#page2">second page</a>.</p>

   </div>
   <div data-role="footer">
      <h4>mobilexweb.com</h4>
   </div>

   </div>
```

```
    <div data-role="page" id="page2" data-title="This is the second page">

       <div data-role="header">
          <h1>Second page</h1>
       </div>
       <div data-role="content">
          <p>This is the main content of the second page</p>
          <p>You can go back using the header's button, <a href="#page1">
              clicking here</a> or using your browser's back button.
       </div>
       <div data-role="footer">
       <h4>mobilexweb.com</h4>
    </div>

 </div>

 </body>
```

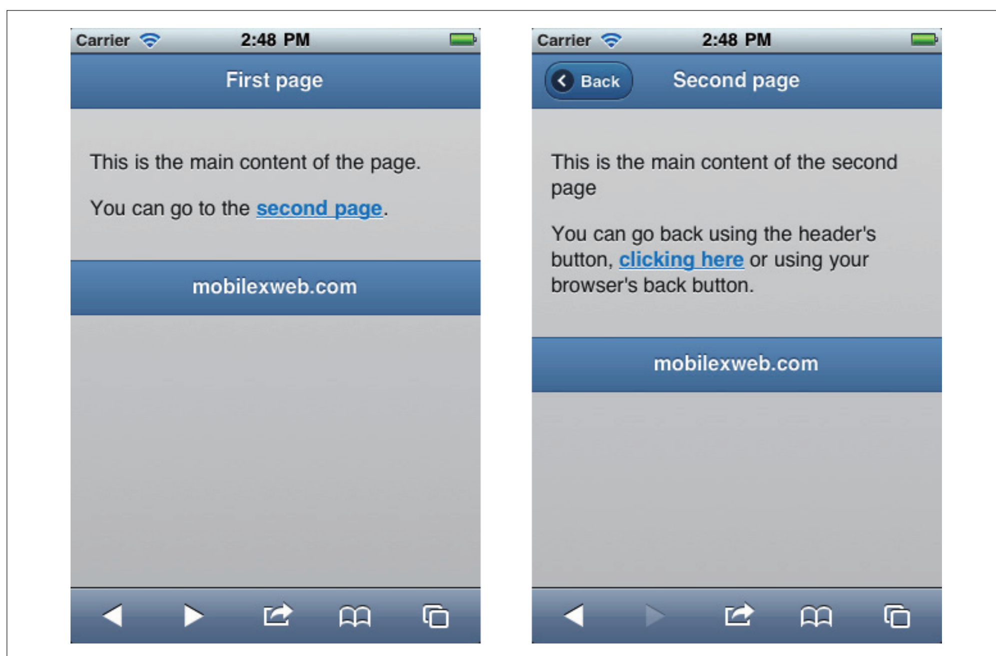


图 2-8：在内部页面之间导航很容易，跳转使用的是标准链接标签

整个 Web 应用可以只放在一个 HTML 文档里吗？如果不需要动态内容，当然是可以的。不过要记住，在一个文档中嵌入多个页面时，即使用户看不见所有页面，他也必须把这个文档完整地下载下来。我们需要在性能与可用性之间找到一个平衡，从而决定在一个文档中应该包含多少页面。

图 2-9 展示了一个内部页面导航的图解。

图 2-9：在内部页面之间导航时，每个向前的链接将在访问历史（一个栈）中添加一条记录，每个后退行为则从中移除最后一条记录

可以在网站上保存状态数据，了解用户可能会访问哪些页面，以便将它们预加载到同一个文档中。



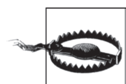
jQuery Mobile 会自动将指向前一个页面的链接的行为转换为后退，点击时会显示一个后退的动画，不会在浏览器访问历史中添加新记录。如果希望一个链接的行为是后退，可以在这个 a 元素上添加 `data-rel="back"` 属性。

2.4.3 外部页面链接

如果不想将一个页面与第一个页面包含在同一个文档中，或者内容需要动态地创建（例如使用 PHP 或其他服务端代码），则可以使用标准的 a 标签链接到另一个 jQuery Mobile HTML 文档。

```
<a href="otherDocument.html">Gotonextpage</a>
```

这就完成了，jQuery Mobile 会自动处理上面这种链接，它的访问体验与内部链接非常类似。



外部页面与当前页面必须在同一个域名下，或者在同一个本地应用的包中。指向其他域名的链接将被当作绝对外部链接处理。

唯一的区别是，jQuery Mobile 会使用 AJAX 方式请求这个文档，解析它的内容，将对应的页面添加到当前 DOM 中，再平滑地过渡到这个新页面。请求正在进行时，用户会看到一个旋转的、表示正在加载的漂亮图标，如图 2-10 所示。记得使用 `data-title`，以便新页面加载后能更新浏览器的标题。

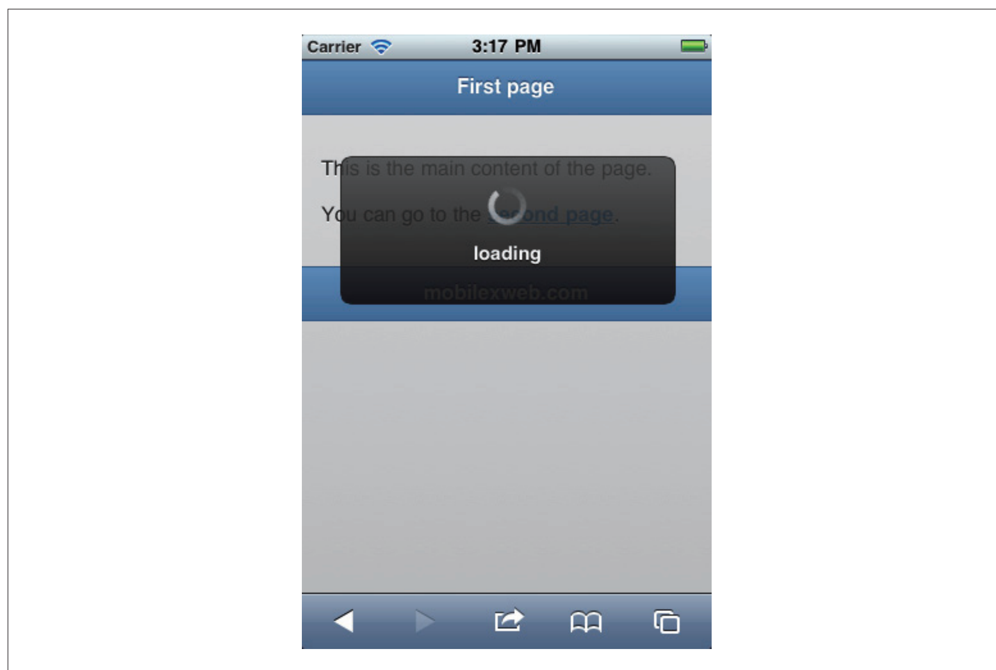


图 2-10：链接到外部 jQuery Mobile 页面时，请求页面加载的时间如果过长，框架会自动显示一个旋转图片的动画

让我们再来梳理一下。jQuery Mobile 在页面上发现一个链接时，它首先判断这个链接是内部链接（以散列字符 # 开始）还是外部链接。如果是个外部链接并且没有定义 rel 或 target 属性（就像我们上面写的例子一样），它将捕获对这个链接的点击事件，向链接的 href 文档发起一个 AJAX 请求，同时显示一个模态的进度窗口。请求完成时，jQuery Mobile 会将加载的页面添加到当前 DOM 中并转向它，就像它是个内部页面一样。



对那些不支持 jQuery Mobile 的浏览器，外部链接将像正常的 HTML 链接一样工作，不会有任何问题，因为作为标准的 a 标签，这本来就是它们的功能。这就是渐进增强的力量。

jQuery Mobile 会自动使用一个新的 ID（由该页面与原始文档的相对 URL 定义）将外部加载的页面添加到当前 DOM 中。因此，如果在这个 jQuery Mobile 文档仍然驻留在浏览器中时用户又访问了这个页面，它将是已经“在台上”的状态。这个操作见图 2-11。

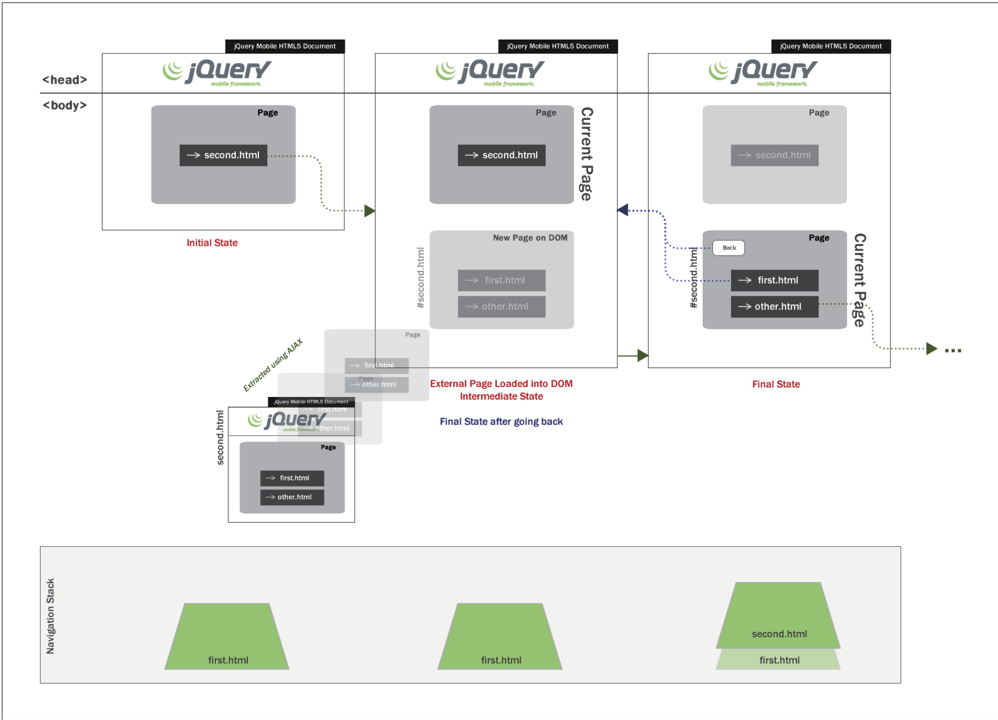


图 2-11：jQuery Mobile 会把通过 AJAX 加载的外部链接页面插入到原始文档的 DOM 中，这样再次访问这个页面时它将处于已预读取状态

URL与jQuery Mobile的导航

在一些设备上，jQuery Mobile使用散列导航（#<id>）来在页面之间移动。因此，在jQuery Mobile文档中，用于在同一个页面中滚动的HTML锚点导航将不能工作。

模拟这个特性需要用JavaScript。在不兼容的设备上，内部页面会同时全部显示，内部链接则会像普通的HTML锚点导航链接那样工作。

有一些移动浏览器支持访问历史API（HTML5新API之一），该API允许网站在不刷新整个页面的情况下改变整个URL。jQuery Mobile会在支持的浏览器上使用这个API，在这些浏览器上访问外部链接时，即使内容是通过AJAX加载的，浏览器的URL地址栏仍然会被改变为完整的目标URL，而不是仅仅改变散列导航#<id>。

有很多种原因可能导致 AJAX 请求失败，如服务器错误、URL 找不到、连接失败，或者设备的网络问题。如果无法加载目标页面，用户将看到一个类似图 2-12 中所示的错误警告。在本书后面的章节我们将学习如何定制这个错误消息。

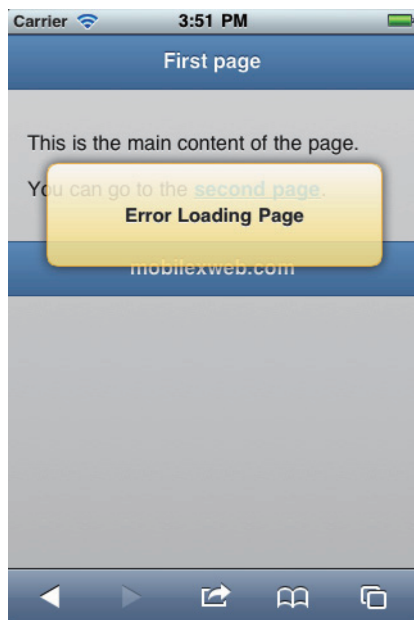


图 2-12：外部 jQuery 页面无法加载时向用户显示的消息

AJAX还是Hijax？

AJAX（本意为异步JavaScript与XML）是一种客户端技术，允许浏览器在不改变当前页面的URL也不阻塞页面UI的情况下，在后台创建一个到服务器的请求。最初，AJAX请求的数据格式为XML，但现在更常见的情况是传输JSON格式的值，甚至普通文本或HTML。

jQuery Mobile在页面导航之间创建AJAX请求，并将整个目标HTML文档作为留待后面解析的普通文本，同时，在文档源码中仍然有普通的HTML链接。这种模式被称为Hijax技术[1]，它构建在渐进增强技术之上。

要将外部页面链接到一个页面中而不出问题，需遵守以下规则。

- 目标也必须是一个 jQuery Mobile 文档。
- 目标必须与当前页面在同一个域名下。
- 目标必须是一个只包含一个页面的文档。
- 如果目标 URL 是一个文件夹（如 /clients），则链接的 `href` 属性必须以斜杠结尾，即 `href="/clients/"`。
- 不能定义 `target` 属性，如 `target="_blank"`。



jQuery Mobile Web 应用的生命周期一般会包括一个完整的 HTTP 请求（第一个文档），可能要加载许多内部页面，以及可能有许多对外部页面的 AJAX 请求。除非遇到绝对外部链接，或者是在 B 级或 C 级浏览器上，不然不会再有完整的 HTTP 请求。

为了避免与框架的逻辑冲突，需要将包含多个页面的外部文档当作绝对外部链接处理，下一节我们会讨论这个问题。

jQuery Mobile 只会从加载的文档中取出第一个页面（第一个带有 `role="page"` 的 `div`），其他内容都将被忽略。也就是说，所有目标文档的 head 元素中的信息以及其他在首个页面元素之外的内容都将被忽略。

记住，其他文档中嵌入的任何 CSS 或 JavaScript，和 `title` 元素一样，都将被忽略。那该把它们删除吗？绝对不要。别忘了 jQuery Mobile 使用渐进增强技术，在那些不兼容的设备上，这些被忽略的内容将用于在浏览器中加载完整文档。稍后我们会讨论如何处理这个限制。

译注 1：hijax 与 hijack 谐音，hijack 的意思是“劫持”，因此 Hijax 可理解为“拦劫点击操作，发送 Ajax 请求”。

预读并缓存页面

有时我们想在用户选中选项时立即显示下一个页面，因此不想使用外部页面。为了减少AJAX的时间，我们可以预读一个页面。我们可以告诉jQuery Mobile框架哪些链接需要预读，这样这些页面就会在DOM中更快就绪。要实现这一点，只需为链接加上一个布尔属性data-prefetch，例如：

```
<a href='newpage.html' data-prefetch>转到这个新页面</a>
```

记住这个特性应该只在那些很有可能被点击的链接上使用。一些好的用例如：分页的文章、照片画廊（预读下一张图片）或者统计结果显示大多数用户会点击的链接。这个特性会增加用户的HTTP流量及消费，需谨慎使用。

为了避免DOM占用过多内存，当外部加载页面变得不可见时（在后退或前进到一个新页面后），jQuery Mobile会自动从DOM中将它移除。要再次显示这个页面时，jQuery Mobile会尝试从缓存中恢复它，如果失败则从服务器重新下载。如果想强制指定一个外部页面不从DOM中移除，可以在页面元素上指定属性data-dom-cache="true"。由于DOM的增长经常会导致内存及性能的问题，因此请仅在确信用户会再次访问时才使用这个特性。

2.4.4 绝对外部链接

有时我们需要链接到另外的非 jQuery Mobile 内容的站点或文档，这时就需要显式定义一个绝对外部链接。可以通过在 a 标签上添加 data-rel="external" 来实现。

例如：

```
<a href="http://www.mobilexweb.com" data-rel="external">查看我的博客</a>
```

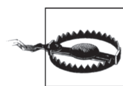
还有一些情况下的链接，比如指定了 target 属性或链接到不同域名的文档，无论这些文档是不是 jQuery Mobile 文档，也都会被当作绝对外部链接处理：

```
<a href="http://www.mobilexweb.com" target="_blank">查看我的博客</a>

<a href="http://www.otherdomain.com/whatever">查看我的博客</a>
```

另一个强制将链接转为绝对外部链接的方法是在链接中使用 data-ajax=false 属性（对同一个域名下的页面很有用）：

```
<a href="otherpage.html" data-ajax="false">别的页面</a>
```



用户点击（或用他的手指轻击）一个绝对链接时，jQuery Mobile 实例将被卸载，浏览器将转向到指定目标。如果想在打开链接的同时保持 jQuery Mobile 实例，则应该在超链接中加上 `target="_blank"`，大多数支持多页面浏览的智能手机及平板电脑都支持这个属性。

创建本地 Web 应用时对外部链接要三思而后用。默认情况下，最后的目标页面会在你的 Web 应用容器内（在你的应用的限制下）打开，这可能并不是你所期望的。例如，如果你的本地应用没有提供后退等导航按钮的话，用户可能就没法再回到你的应用了。有一些解决方案，如 PhoneGap，会将这个 URL 在默认的浏览器中打开，同时关闭或最小化你的应用。



如果是链接到一个绝对外部 URL 的 jQuery Mobile 文档，一切都会正常，不过目标页面将没有后退按钮。目标页面将初始化一个新的 jQuery Mobile 实例。

后面的章节我们会讲到使用 jQuery Mobile 开发本地应用，在此之前，如何解决这个问题先留给你思考。

2.4.5　移动互联网特有链接

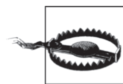
不要忘了我们正在创建的是移动互联网体验，应该将应用与设备尽可能整合，以便提升用户体验。通过 URI 机制，我们可以拨打电话或发送短信，我们将在 2.6 节深入讲解这个功能。



记住对你的网站应用渐进增强模式。这意味着应该总是使用标准的 a 链接，而不要使用 JavaScript 来改变页面或导航。使用标准 HTML 导航将让你的 Web 应用在所有移动浏览器场景中都能工作。

2.4.6　页面间的过渡效果

就像之前看见的，用户从一个页面转到另一个页面时，jQuery Mobile 会使用一个平滑的动画过渡。默认情况下，所有的过渡都使用从右到左的动画，这种动画模式在从主干到细节或普通到具体的导航时非常有用。在导航中深入前进时，会看到从右到左的动画，返回到之前的页面时，则会看到反转效果：从左到右的动画。



过渡效果使用了 CSS3，大多数设备上它们都已被硬件加速。在一些不兼容的设备上，页面切换时将没有动画。

jQuery Mobile 允许我们为每个页面切换指定使用的动画过渡效果。要实现这一点，需要在链接中使用自定义属性 data-transition。每个过渡效果都有一个对应的用于后退的相反的效果。

可用的过渡效果如下（参见图 2-13）。

- slide
  默认的从右到左的动画。

- slideup
  从下到上的动画，多用于模态页面。

- slidedown
  从上到下的动画。

- pop
  新页面会从屏幕中间的一个小点开始变大，直到占满屏幕。

- fade
  老页面与新页面交叉淡入淡出渐变动画。

- flip
  一个 2D 或 3D 旋转动画。3D 只在一部分设备上可用，如 iOS 设备。在其他如基于 Android 的设备上，这个过渡效果将渲染为 2D 旋转，这或许并不是你所期望的效果。

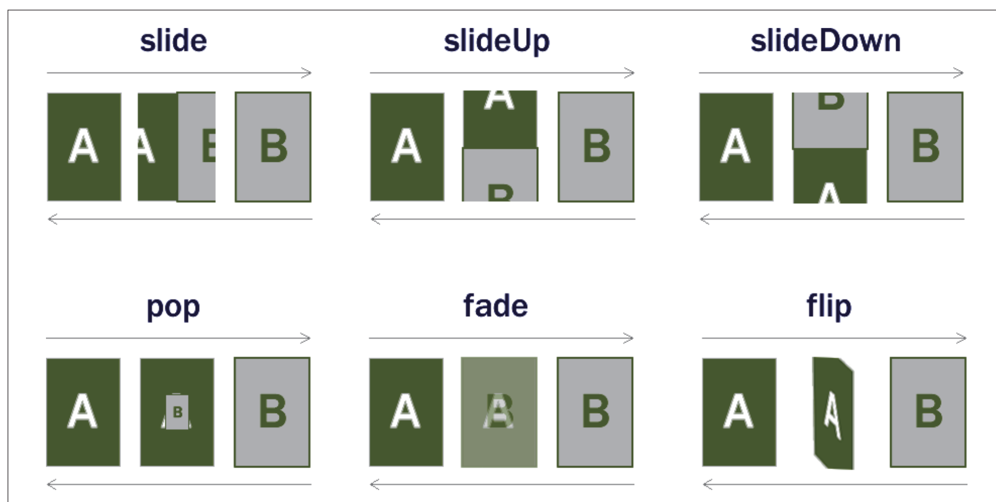


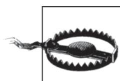
图 2-13：jQuery Mobile 1.0 中支持的页面过渡效果



jQuery Mobile 1.0 中的大多数过渡效果由 jQTouch（http://jqtouch.com）团队开发，由 David Kaneda 创建，现在由 Jonathan Stark 维护。自 1.1 版开始，jQuery Mobile 正在改进过渡效果，另外还添加了两个新值：flow 和 turn。

要在链接上定义一个新的过渡效果，只需使用 data-transition="name"，例如：

```
<a href="#page2" data-transition="pop">第二个页面</a>
```



过渡效果动画只在 A 级浏览器上的内部链接或连到 jQuery Mobile 页面的外部链接上生效。对于绝对链接或移动互联网专有 URI 方案（如发送短信），这些动画将失效。

建议在页面间使用经典的过渡效果，同时保持一致的 UI 心理模式。例如，在内容中深入或层级导航时，总是使用 slide 过渡。对帮助、设置以及添加新项等平级操作，则应该使用其他过渡。

记住，切换效果是应用到链接上而不是页面上的。这意味着如果有多个链接指向一个页面的话，转到这个页面时可以有不同的过渡效果。在 Web 应用中，这种情况应该尽量保持过渡效果的一致性。

2.4.7 反转过渡效果

可以在链接上指定 data-direction="reverse" 来强制使用反转过渡效果（后退行为），这时 jQuery Mobile 将自动反转指定的过渡效果。

2.5 对话框

*对话框*是在 Web 应用中显示页面的另一种方式，它并不是新东西，只是使用了不同语义的页面。

对话框页面用于显示模态消息、列表，或与原页面没有层级关系的信息。

对话框页面与普通页面的最大区别如下（见图 2-14）。

- 如果指定了 data-add-back-btn，则在左上角后退操作按钮的位置将显示关闭操作按钮（带有一个叉）。
- 显示的内容带有边距，不是全屏页面，而是在原页面上弹出的窗口。
- 不会在访问历史中留下记录。（如图 2-15 所示，在对话框中打开新页面时，新页面会像对话框不存在一样打开。外部页面的行为也是这样。）

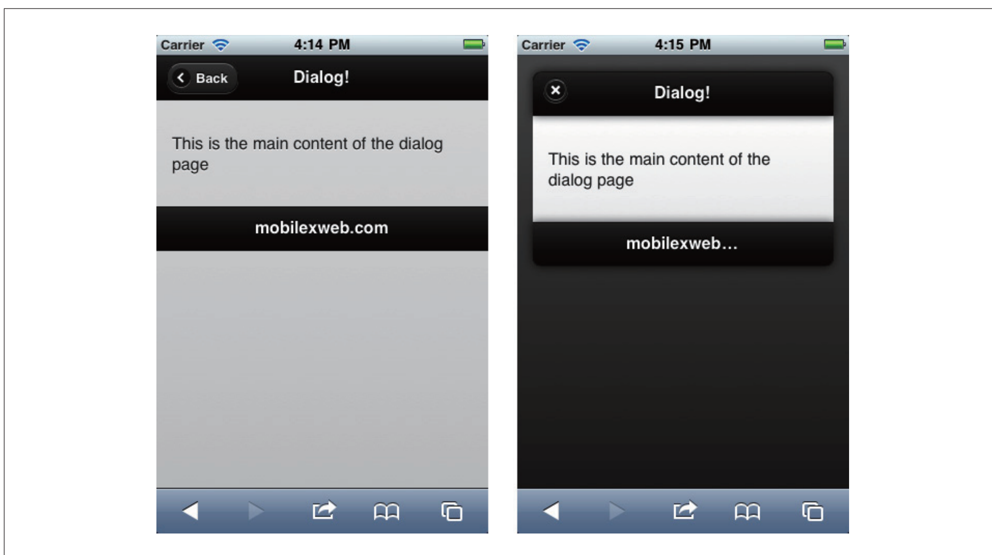


图 2-14：同一个页面正常打开和以对话方式打开的效果

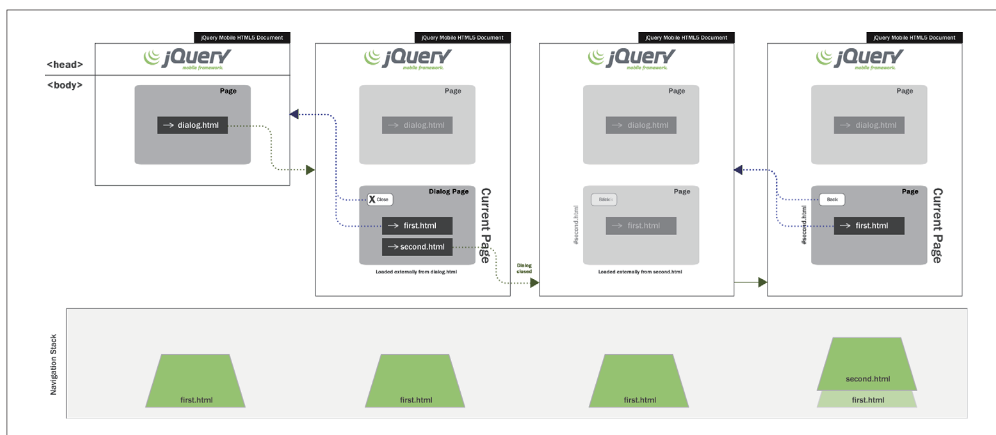


图 2-15：对话框页面不会增加访问历史记录

要打开对话框页面，只需在链接的 a 标签上加上 data-rel="dialog" 属性。这个 rel 定义了当前页面与所链接的页面之间的关系（这儿是对话关系）。例如：

```
<!-- 这个链接指向一个外部页面，将显示为对话框 -->
<a href="./confirmation.html" data-rel="dialog">删除此项</a>
```

为了避免引起用户迷惑，最好将对话框的默认过渡动画改为 slide 之外的效果，以便与层级页面之间的标准动画区别开来。因此，一个典型的打开对话框的链接看起来会像这样：

```
<!-- 这个链接指向一个外部页面，将显示为对话框 -->
<a href="./confirmation.html" data-rel="dialog" data-transition="pop">
删除此项 </a>
```

页面被用作对话框时，可以通过 `data-overlay-theme` 指定覆盖层的色卡。



除了在链接上使用 `data-rel="dialog"` 属性之外，还可以使用 `data-role="dialog"` 而不是 `data-role="page"` 来生成对话框。

2.5.1 关闭，还是后退

如图 2-14 所示，以对话框方式打开的页面不能后退，只能关闭。这意味着使用对话框时 jQuery Mobile 内部的导航处理有一些改变。

第一个改变是对话框“属于”打开它的页面，不会在访问历史中增加新的记录。

对话框页面和典型的桌面应用中的弹出框或模态内容的行为很类似。使用链接、按钮或其他 UI 控件关闭对话时，只应该链接回原来的页面。jQuery Mobile 会将指向原页面的链接当作关闭操作处理。



后几章将分析如何使用主题及 CSS 自定义页面、对话框以及组件的视觉设计。

假设在 delete.html 上有一个删除操作，这个链接会打开一个对话框（confirm.html）以便让用户确认删除。在确认对话框上，应该有两个操作：删除和取消。

对话框页面不会在导航历史中添加记录。因此，如果用户在对话框开着时刷新页面，他将看到对话处于关闭状态的原始页面。

取消链接应该关闭对话，和左上角的叉形按钮一样。这时，取消链接应该只是一个普通的指向 delete.html 的 `a` 标签。

为了让这个例子更完善一些，这儿引入一点下一章的内容。在链接上使用 `data-role="button"`，这些下划线链接将被转成全屏宽度、适合手指操作高度的按钮。后几章将讨论更多关于按钮以及如何定制它们的内容。



下一章我们将看到一种使用 JavaScript 来关闭对话的方法。不过，如果我们的 Web 应用的目标包括不兼容 jQuery Mobile 的设备的话，推荐使用标准的链接来返回打开它的页面。记住 jQuery Mobile 使用渐进增强，即使浏览器不支持 JavaScript，Web 应用仍能工作。

图 2-16 展示了对话框的实际效果。

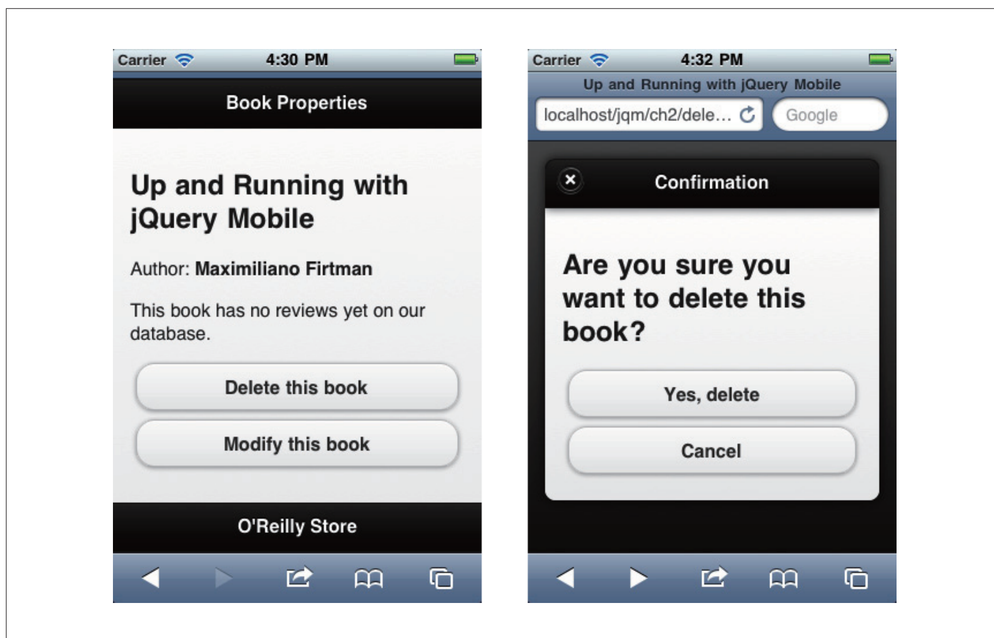


图 2-16：在 iPhone 上的对话示例

下面是 delete.html 的代码：

```
<!DOCTYPE html>
<html>

<head>
 <meta charset="utf-8" />
 <title>Up and Running with jQuery Mobile</title>
 <link rel="stylesheet" href="jquery.mobile-1.0.min.css" />
 <script src="jquery-1.6.4.min.js"></script>
 <script type="text/javascript" src="jquery.mobile-1.0.min.js">
 </script>
 <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body>

 <div data-role="page" id="main">

   <div data-role="header">
     <h1>Book Properties</h1>
   </div>

   <div data-role="content">
```

```
          <h2>Up and Running with jQuery Mobile</h2>
       <p>Author: <b>Maximiliano Firtman</b></p>
       <p>This book has no reviews yet on our database.</p>

          <p>
           <a href="./confirmation.html" data-rel="dialog" data-transition=
               "pop" data-role="button">
           Delete this book
           </a>

           <a href="" data-role="button">
            Modify this book
           </a>
       </p>
     </div>

     <div data-role="footer">
       <h4>O'Reilly Store</h4>
     </div>

   </div>

  </body>
 </html>
```

下面是 confirmation.html 的代码：

```
<!DOCTYPE html>
<html>

<head>
<meta charset="utf-8" />
 <title>Delete Confirmation</title>
 <link rel="stylesheet" href="jquery.mobile-1.0.min.css" />
 <script src="jquery-1.6.4.min.js"></script>
 <script type="text/javascript" src="jquery.mobile-1.0.min.js"></script>
 <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body>

 <div data-role="page" id="alert">
   <div data-role="header">
     <h1>Confirmation</h1>
   </div>
   <div data-role="content">
     <h2>Are you sure you want to delete this book?</h2>

     <!-- 这将是一个正常的页面加载 -->
        <a href="delete-yes.html" data-role="button">Yes, delete</a>

     <!-- 这只是一个关闭链接 -->
```

```
      <a href="delete.html" data-role="button">Cancel</a>
    </div>

  </div>

  </body>
</html>
```



可以在页面上使用 data-close-btn-text 属性来自定义关闭按钮的文本。

2.5.2　从对话框打开页面

对话框可以有指向其他页面的普通链接（或绝对外部链接）。用户点击了指向非原页面（如上一节所述）的链接时，jQuery Mobile 会关闭对话，回到原页面，然后打开新页面，就像这个链接本来就在原页面上一样。

在我们的例子中，如果点击了“Yes, delete”按钮，则对话框将被关闭，接着是一个典型的过渡效果，然后到达 delete-yes.html 页面。

图 2-15 中的图表显示了这种情况下导航发生了什么。

2.6　与电话整合

来，跟我重复一遍：“我们正在为移动设备设计页面，移动设备，移动设备。”这意味着什么？意味着我们应该尽可能地与电话整合。

实现这一点的一个方法是通过 URI 机制，a 标签链接的 href 属性中可以使用不同的协议。相信你已经对 mailto: 协议很熟悉了，但在移动浏览器上，还有很多协议。这些协议有些是大多数设备都支持的，还有些则是依赖平台的（如 iOS）。依赖平台的协议在创建本地 Web 应用并知道它会在什么平台上运行时很有用。

2.6.1　拨打电话

记住：大多数移动设备同时也是电话！所以，为什么不能创建拨打电话的链接呢？如果你正在创建业务指南，即使用的是自己熟悉的手机，大多数人都宁可打电话而不是在设备上填写表单。图 2-17 显示了在不同的设备上拨打电话的情况。

首选的方法（日本 i-Mode 标准）是使用 tel:< 电话号码 > 方案：

```
<a href="tel:+1800229933">免费呼叫我们！ </a>
```

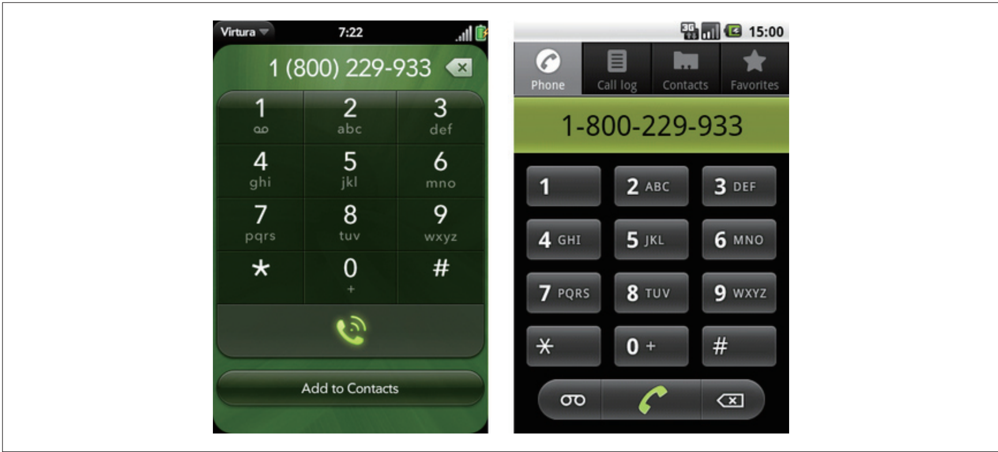


图 2-17：点击电话链接时，Palm 的 webOS 及 Android 会显示呼叫窗口

这个 URI 方案在 RFC5341（http://tools.ietf.org/html/rfc5341）中被提议为标准，不过请仔细阅读这份标准，因为该提议的大部分参数现在还没有设备支持。



建议插入的电话号码使用国际格式：加号（+），国家代码，区号，最后是本地号码。我们无法预料访客来自何处，即使他们来自本国，甚至来自本区，国际格式的电话号码也没有问题。

用户激活电话链接时将看到一个确认警告，询问他是否要拨打这个电话，警告上会显示完整的电话号码以便用户做出决定。这是为了避免欺骗用户拨打国际电话或收费电话。

一些非电话移动设备（如 iPod Touch 或 iPad）不允许语音呼叫，但它们会提示是否将该电话号码添加到电话簿中（见图 2-18）。

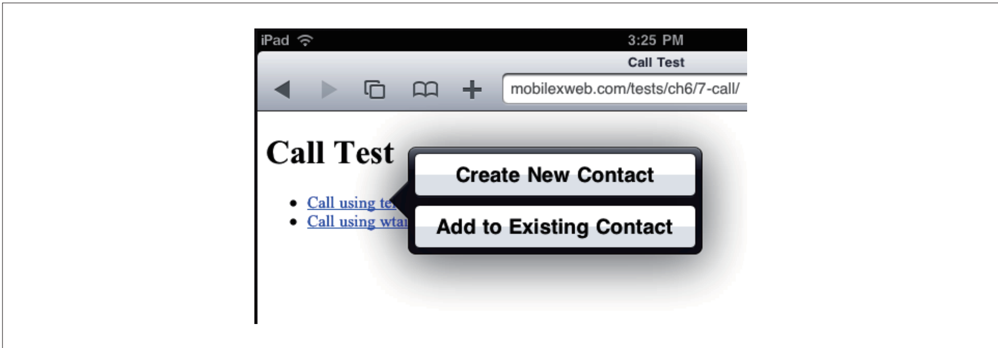


图 2-18：一些设备（如苹果的 iPad）不能打电话，不过它们为电话链接提供了别的选项

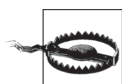
2.6.2 视频及VoIP呼叫

基于 iOS 的带摄像头的设备，如 iPhone 4 及 iPod Touch 4G，包含一个名为 FaceTime 的视频聊天应用。为此类设备创建的应用可以使用 `facetime://<用户名或号码>` 来创建视频呼叫。在别的设备上点击这种链接会出错：

```
<a href="facetime:101010">用 Facetime 呼叫我 </a>
```

Skype 也有它专用的链接 URI 机制。要创建这种链接需要 Skype 用户名。还可以添加一个可选的 `?call` 参数来立即初始化一个呼叫，如果不加这个参数，我们会看到用户的个人信息：

```
<a href="skype:skype_user?call">用 Skype 呼叫我们 </a>
```



如果设备上没有安装链接中指定的本地应用，比如 FaceTime 或 Skype，则用户将收到一个错误。在不支持的设备上，最好不要显示这些链接。关于特定设备及内容自适应相关的主题，可参见《移动网络程序设计》（O'Reilly）一书。

关于移动浏览器的 URI 方案的更新信息，可访问 http://www.mobilexweb.com/go/uri。

2.6.3 发送电子邮件

一些带有浏览器的现代设备也有能响应传统互联网的 `mailto:` 协议的邮件应用。语法是 `?parameters`。不同设备的参数可能不同，但基本上包括 `cc`、`bcc`、`subject` 以及 `body`。参数以 URL 格式（`key=value&key=value`）定义，值必须已经过 URI 编码。

下面是一些例子：

```
<a href="mailto:info@mobilexweb.com">给我们发邮件 </a>
<a href="mailto:info@mobilexweb.com?subject=Contact%20from%20mobile">
给我们发邮件 </a>
<a href="mailto:info@mobilexweb.com?subject=Contact&body=This%20is%20
the%20body">给我们发邮件 </a>
```

记住 `mailto:` 机制并不能保证消息被发出。它一般只是打开邮件应用，在做了可能的修改之后用户必须自己确认邮件是否已被发送。如果需要确保邮件被发送，可以使用服务器端机制。

一般来说，如果想在邮件的正文中插入新行，可以使用回车加换行符（%0D%0A）。在 iOS 的邮件应用中这个方法不管用，不过可以在 `body` 中插入 HTML 标签，因此，可以在移动版 Safari 浏览器中使用 `<br/>`：

```
<a href="mailto:info@mobilexweb.com?subject=Contact&body=This%20is%20the%20
    body%0D%0AThis%20is%20a%20new%20line">给我们发邮件</a>

<a href="mailto:info@mobilexweb.com?subject=Contact&body=This%20is%20the%
    20body<br/>
This%20is%20a%20new%20line">从iPhone上给我们发邮件</a>
```

2.6.4　发短消息

我们都喜欢短消息服务，因此移动浏览器一般都提供了通过链接打开新短消息窗口的功能。要实现这一点，有两种可能的URI方案，分别为sms://和smsto://。不幸的是，没有标准的检查用户浏览器支持哪种方案的方法。不过，对只使用jQuery Mobile的智能手机，可以安全地使用sms://。

发送短消息的语法是sms[to]://[<目标号码>][?参数]。如你所见，目标号码是可选的，也就是说不带任何参数即可打开设备的短消息编辑窗口。出于安全原因（避免收费短消息内容），不是所有电话都支持定义消息内容参数。和发电子邮件一样，用户点击链接时，不会自动发送短消息，只是打开编辑短消息的窗口，用户必须手动完成整个过程。目标号码应该要么是个国际号码，要么是个短号码，在后一种情况下，我们必须确保用户在既定的国家，并且已连接到一个支持那个短号码的运营商。

这儿是一些例子：

```
<a href="sms://">发送短消息</a>

<a href="sms://?body=Visit%20the%20best%20site%20at%20http://mobilexweb.
    com">通过短消息邀请朋友<a>

<a href="sms://+3490322111">通过短消息联系我们</a>

<a href="sms://+3490322111?body=Interested%20in%20Product%20AA2">关于产品
AA2的更多信息</a>
```

2.6.5　其他URI方案

如果想深入了解HTML与电话的整合，包括彩信、iDEN网络直接呼叫（iDEN-networks Direct Call）、视频通话、黑莓PIN消息、Facebook、Twitter以及其他原生应用的整合，可以阅读《移动网络程序设计》一书。



专有URI方案不是jQuery Mobile的功能。jQuery Mobile只是将URI保持原样，移动浏览器会响应它们。

2.6.6 综合起来

让我们创建一个简单的使用不同 URI 方案的 jQuery Mobile Web 应用：

```
<!DOCTYPE html>
<html>

<head>
 <meta charset="utf-8" />
 <title>jQuery Mobile 深入浅出 </title>
 <link rel="stylesheet" href="jquery.mobile-1.0.min.css" />
 <script src="jquery-1.6.4.min.js"></script>
 <script type="text/javascript" src="jquery.mobile-1.0.min.js"></script>
 <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body>

<div data-role="page" id="main">
   <div data-role="header">
     <h1> 专有链接 </h1>
   </div>
   <div data-role="content">
     <p> 点击下面的按钮测试移动设备专有链接的行为。</p>

        <p>
         <a href="tel:+1800229933" data-role="button">
          呼叫白宫
         </a>

         <a href="sms:+1800229933" data-role="button">
          给白宫发短消息
         </a>

         <a href="sms:+1800229933?body=Hello!" data-role="button">
          带正文的短消息
         </a>

         <a href="mailto:info@mobilexweb.com?subject=Sent%20from%20the%20web"
             data-role="button">
          给我发邮件
         </a>

         <a href="skype:maximiliano.firtman?call" data-role="button">
          用 Skype 联系我
         </a>

         <a href="facetime:+1800229933" data-role="button">
          用 Facetime 呼叫我
         </a>
          (仅限带摄像头的 iOS 设备)
    </p>
  </div>
  <div data-role="footer">
    <h4>www.mobilexweb.com</h4>
  </div>
 </div>

</body>
</html>
```

第 3 章

UI组件

jQuery Mobile 为 Web 应用准备了相当多的 UI 组件。记住，我们总是可以使用普通的 HTML 和 CSS 来添加内容及实现各种创意。不过，为了兼容各个平台，我们将优先使用框架提供的各种组件。

jQuery Mobile 组件可以分成以下几类：

- 工具栏组件；
- 格式化组件；
- 按钮组件；
- 列表组件；
- 表单组件。

本章将讲解前三类组件，列表及表单组件将放到后面的章节讲解。

3.1 工具栏

工具栏在 Web 应用中是可选的，用于定义页头以及 / 或者页脚的区域。页头及页脚都是可选的，不过基本上每个移动 Web 应用都有页头。

之前的章节中我们已经讨论过页头，它是页面顶部用于放置标题以及 / 或者后退 / 关闭按钮的条形区域。页头通常由一个 `role` 为 `header` 的 `div` 定义，其中包含一个 `h1` 标题。

```
<div data-role="header">
   <h1> 页面标题 </h1>
</div>
```

页脚与页头类似，位于 Web 应用的底部，用途更为广泛。它可以包含版权信息、呼叫（call action）链接，或者工具栏或标签导航在内的一系列按钮。页脚通常是一个 role 为 footer 的 div：

```
<div data-role="footer">

</div>
```

3.1.1 定位

定位工具栏看起来很容易：页头在顶部，页脚在底部。不过，jQuery Mobile 的定位系统让我们可以为这两个工具栏定义不同的行为。

每个工具栏（页头或页脚）都可以用四种方法定位：

- 内联模式；
- 标准固定模式；
- 全屏固定模式；
- 真实固定模式。

内联模式是各个工具栏的默认模式。这意味着页头和页脚会在页面上以内联流的方式呈现，也就是说如果页面的内容长度超出可见高度时页脚将被隐藏，直到向下拖动滚动条，同时页头只在滚动条在顶部时才可见。

有时需要让两个工具栏总是可见，因此 jQuery Mobile 提供了固定以及全屏模式。可以使用 data-position 属性来定义工具栏的定位模式。

jQuery Mobile 中的固定工具栏会位于页面顶部（页头）或底部（页脚）。用户滚动页面内容时，工具栏会自动淡出隐藏，滚动完成时，固定工具栏会再次在顶部或底部对应的位置出现。

用户点击页面上某个非交互的内容区域时，固定模式的工具栏会转换为内联模式，反之亦然。这个机制允许用户通过触摸屏幕在内联 / 固定模式之间方便地切换，以便在需要时有更多的空间来显示内容。

如图 3-1 所示，要定义一个固定工具栏，只需使用 data-position="fixed"。

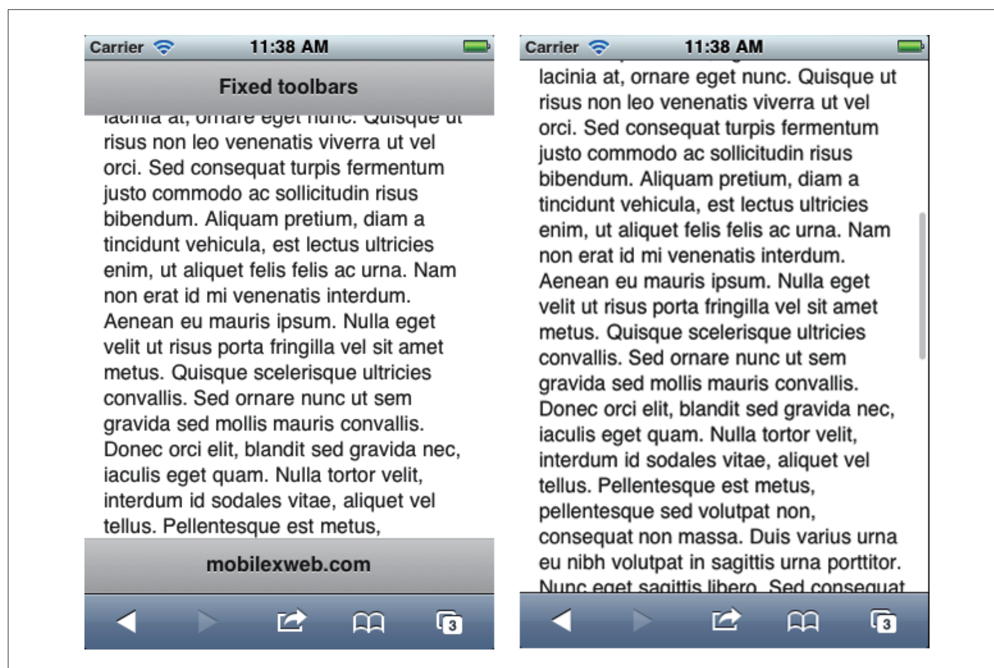


图 3-1：无论页面内容有多长以及初始状态下滚动条的位置在哪儿，固定工具栏都将位于页面顶部（用于页头）或底部（用于页脚），不过页面滚动时，工具栏将被隐藏

全屏工具栏其实是一个只在用户点击屏幕时才出现的隐藏的固定工具栏，用户再次点击屏幕时，它又会消失。使用这种工具栏时内容将全屏显示，用户需要工具栏时，只需在内容上点击一下。



对固定及全屏工具栏来说，用户都可以通过点击内容区域来显示或隐藏固定工具栏，差别在于对全屏模式来说，工具栏会真正被隐藏起来，而对固定模式来说，工具栏会转为内联定位。

在显示照片、长文本或大表单时这个特性特别有用，因为可以获得全屏浏览的好处。

工具栏可见时，内容会被遮住。在大多数浏览器中页头会使用半透明，可以透过页头看到后面的内容。

要使用这种模式，只需在工具栏上定义 `data-position="fixed"` 并在当前页面（`data-role` 属性值为 `page` 的元素）上定义 `data-fullscreen="true"`。

下面的例子定义了一个全屏工具栏：

```
<div data-role="page" data-fullscreen="true">

 <div data-role="header" data-position="fixed">
  <h1>页面标题</h1>
  </div>

  <div data-role="content">

  </div>

  <div data-role="footer" data-position="fixed">

  </div>

</div>
```

3.1.2　真实固定工具栏

写作本书的时候，只有一部分移动浏览器支持 CSS 特性 `position:fixed` 以及 `overflow:auto`。固定定位和滚动区域允许我们定义屏幕上的固定区域，即使用户滚动文档，这些区域的位置仍然可以保持不变。视口以及缩放通常会让固定定位的设计变得复杂。

一些移动浏览器的最新版本，包括 iOS 上的 Safari，Android、BlackBerry 以及诺基亚的浏览器，都支持 `position:fixed` 以及 / 或者使用 `overflow:auto` 来在一个区块中滚动的功能。

想提供真实固定工具栏体验，可以开启 jQuery Mobile 中的触摸溢出（overflow）功能。使用 CSS 的 `overflow:auto`，框架将创建一个较小的滚动区域来替代全屏，这样工具栏就可以总是位于一个位置。在 jQuery Mobile 1.0 版中，这个功能默认被关闭了，可以使用 JavaScript 开启它。开启这个特性后，不支持它的浏览器会降级为标准固定工具栏。jQuery Mobile 将来的版本会废弃这个属性，并在支持的设备上默认使用真实固定工具栏。

本书后面的章节会介绍 JavaScript API，现在只需要知道，开启真实固定工具栏的代码是：

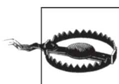
```
$.mobile.touchOverflowEnabled = true;
```

3.1.3　在页头中添加内容

标准的 jQuery Mobile 页头包含一个标题、一个 `hX` 子元素，以及可选的一两个位于页头右侧以及 / 或者左侧的按钮。也可以按需要定制页头的外观及内容。

1. 添加按钮

触摸设备的移动可用性设计通常（但并不总是）在右侧使用肯定动作，在左侧使用否定动作。肯定动作的例子有：完成、保存、是、好以及发送。否定动作则有：取消、返回、不、退出以及注销。



第 2 章我们讲到不需要自己提供后退操作，jQuery Mobile 会自动处理浏览器的后退按钮，也可以在页面上使用 `data-add-back-btn="true"` 来在标题左侧添加一个按钮，如果使用了这个选项，则不应该再在左侧添加其他按钮。

页头的按钮只是一个超链接，使用的是位于页头中的 `a` 元素。如果只提供了一个 `a` 元素，则它将位于标题的左侧。如果添加了两个按钮，则第一个将位于标题左侧，第二个则将位于标题右侧。



如果想强制指定按钮的位置，可以使用 CSS 类：`class="ui-btn-right"` 或 `class="ui-btn-left"`。

下面的页头将只有一个按钮，如图 3-2 所示：

```
<div data-role="header">
   <a href="logout">Log out</a>
   <h1>Title</h1>
</div>
```

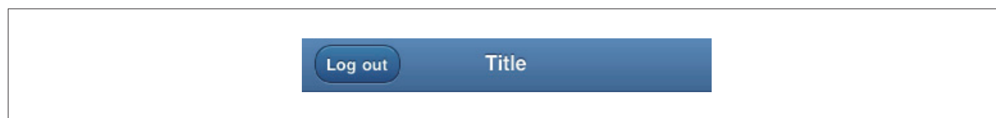


图 3-2：页头可以在标题的左侧或右侧包含按钮

下面的页头将显示两个按钮，右边那个带有图标：

```
<div data-role="header">
    <a href="logout">Log out</a>
    <h1>Title</h1>
    <a href="settings" data-icon="gear">Settings</a>
</div>
```

上例结果见图 3-3，从中可以看到，可以使用 `data-icon` 属性将任意标准 jQuery Mobile 图标应用到工具栏图标上，其中 `data-icon` 可用的值上一章中已经提到，本章 3.4 节会再次涉及。

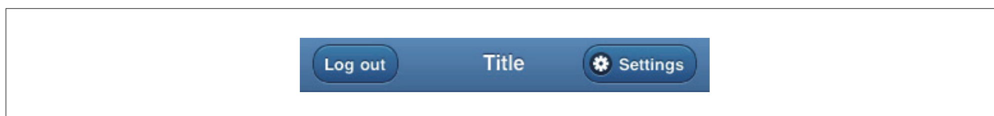


图 3-3：为了提高可读性，工具栏的按钮中经常使用图标

默认情况下，工具栏上的每个按钮都将继承工具栏的主题（如果没有定义，则继承页面主题）。如果想让一个按钮与众不同，可以使用 data-theme 来改变它的色样（参见图 3-4）。

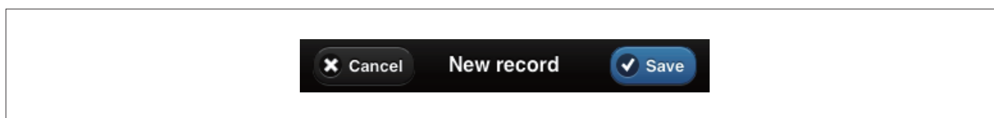


图 3-4：可以使用主题色样改变按钮的外观

例如：

```
<div data-role="header">
   <a href="cancel" data-icon="delete">Cancel</a>
   <h1>New record</h1>
   <a href="save" data-icon="check" data-theme="b">Save</a>
</div>
```



如果想在自定义页头上提供一个指向上一页的按钮，一定要在对应的 a 元素上使用 data-rel="back"，这样 jQuery Mobile 就能正确地处理返回动画以及浏览历史。

2. 添加logo

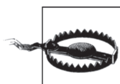
使用图片或 logo 来替代文本是很常见的需求。讲解图片在移动互联网应用中的用法超出了本书的范围，不过如果想在页头中显示一个图片以及标准的左右按钮的话，最好的方式是在对应的 h1 中使用 img 标签：

```
<div data-role="header">
    <a href="cancel" data-icon="delete">Cancel</a>
    <h1><img src="images/logo.png"></h1>
    <a href="save" data-icon="check" data-theme="b">Save</a>
</div>
```

页头会自动适应图片的高度，如图 3-5 所示。记得图片要与当前色样的背景色匹配。出于兼容性考虑，不要使用高度超过 125 px 的图片。



图 3-5：在页面标题中使用图片与使用 img 元素一样简单



记住使用 jQuery Mobile 的目标是同时适用不同的屏幕尺寸和分辨率，因此为了包含图片，需要一些特性检测算法以便为每个设备都提供最好的体验。

3. 自定义页头

如果不想要典型的 jQuery Mobile 页头渲染效果，可以禁用自动页头行为（如对 hX 及 a 的处理），方法为将页头的内容放到一个块容器（通常为一个 div 元素）中。



自定义页头可用来添加表单元素，如用于过滤或菜单访问的下拉列表或复选框。这些控件会在第 5 章讨论。

看一个例子（见图 3-6）：

```
<div data-role="header">
   <div>
       <h1>A custom title</h1>
       <a href="#">A non-button link</a>
   </div>
</div>
```

自定义页头需要自己编写 HTML 及 CSS 代码处理。页头的高度会自动适应自定义的内容。

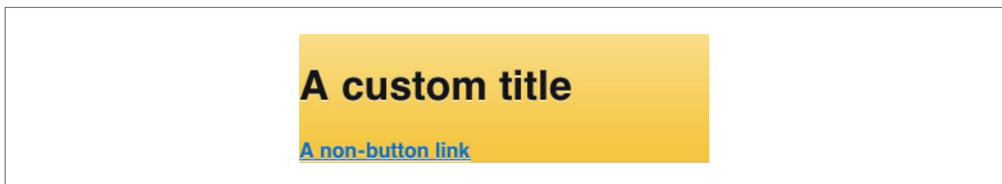


图 3-6：通过自定义页头，可以取消页头中 hX 及 a 元素的自动行为

3.1.4 在页脚中添加内容

页脚远比页头灵活。和页头一样，每个 a 元素都会被渲染为按钮，尽管页脚中没有左、右按钮位。每个按钮都以内联方式添加，一个接着一个。就像在按钮栏中一样，我们可以添加任意数量的按钮。

默认情况下，jQuery Mobile 不会在按钮与页脚边框之间添加内边距。为了让视觉外观更好一些，应该在页脚上添加一个 ui-bar 类。

看一个例子（参见图 3-7）：

```
<div data-role="footer" class="ui-bar">
   <a href="refresh">Refresh</a>
   <a href="filter">Filter</a>
   <a href="search">Search</a>
   <a href="add" data-theme="b">New Item</a>
</div>
```

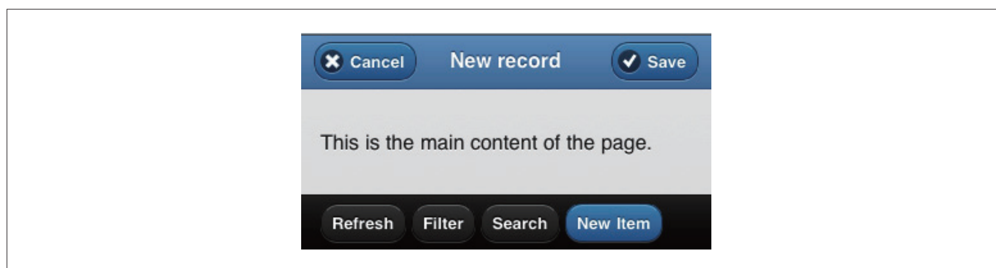


图 3-7：与页头相比，页脚在按钮的数量及排列上更为灵活

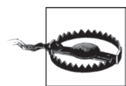


接下来会讲到按钮组，它允许将一系列按钮当作一个大的组合元素处理。这种元素也可以添加到页脚中。

页脚也可以处理表单元素以及导航栏，3.1.5 节将涉及这些内容。

3.1.5 导航栏

导航栏（也称为 navbar）是由一组链接组成的一系列互斥选项，可放置在工具栏上，通常被放在页脚。导航栏是 Web 应用的主要导航方式。在一些平台上，如 iOS 或诺基亚，导航栏也被称为标签栏。



为了避免 UI 错误，导航栏不应该放置动作按钮（如保存、取消以及搜索），而只应该作为 Web 应用的主导航结构。动作按钮应该为页头或页脚上的普通按钮。

导航栏只是一个容器，通常为一个 div 元素，内容为一个包含各个导航链接的无序列表。容器的 role 需要被指定为 navbar：

```
<div data-role="navbar">
   <ul>
    <li><a href="link1">选项 1</a>
```

```
      <!-- ... -->
    <li><a href="linkn">选项 n</a>
  </ul>
</div>
```

如图 3-8 所示，导航按钮与图 3-7 中的标准按钮的外观不同。按钮的宽度根据以下算法计算。

- 导航栏上只有 1 个按钮：100%；
- 导航栏上有 2 个按钮：50%；
- 导航栏上有 3 个、4 个或 5 个按钮：分别为 33%、25% 或 20%；
- 6 个或更多按钮：多行显示，每行 2 个，每个 50%，如图 3-9 所示。

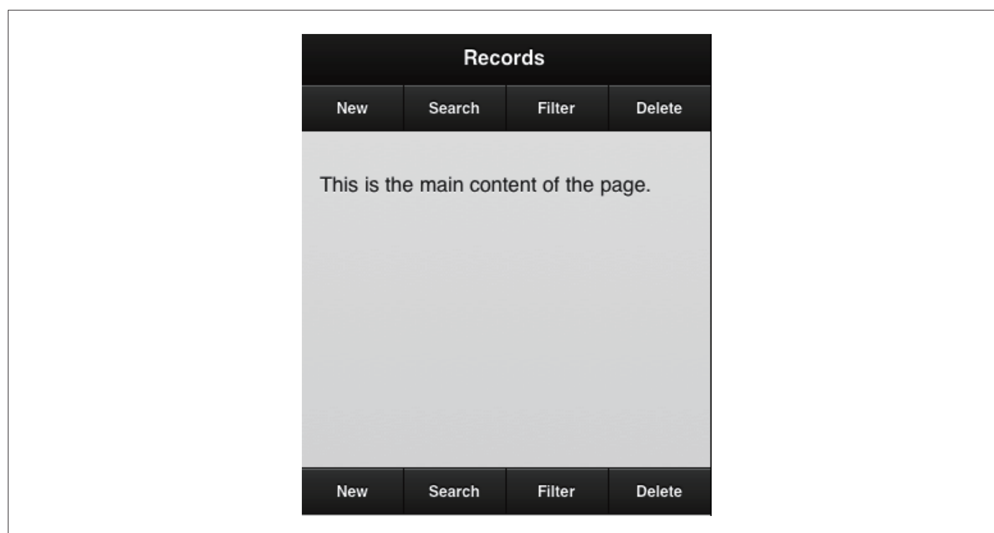


图 3-8：在页头及页脚中的导航栏

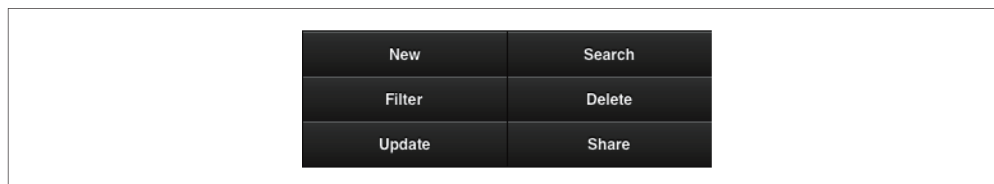


图 3-9：如果按钮数超过 5 个，导航栏排版将改为每行两个按钮



最好将导航栏的元素控制在 6 个以下，以便让它们全部显示在一行。按钮数限制为 5 的原因是在大多数智能手机上，屏幕的五分之一宽度是适合触摸区域的最小推荐宽度。

1. 使用图标

使用标准的 jQuery Mobile 图标机制，可以为每个导航元素添加图标。要实现这一点，可以使用 `data-icon` 属性以及标准的图标名，或者也可以使用自己的图标来自定义。

默认情况下，图标位于文本的顶部，如图 3-10 所示。如果至少有一个导航元素使用了图标，则整个导航栏的高度都会更新以便适应那个带图标的元素：

```
<div data-role="header" data-position="fixed">
  <h1>Home</h1>
  <div data-role="navbar">
    <ul>
      <li><a href="#index" data-icon="home">Home</a>
      <li><a href="#contacts" data-icon="search">Contacts</a>
      <li><a href="#events" data-icon="info">Events</a>
      <li><a href="#news" data-icon="grid">News</a>
    </ul>
  </div>
</div>
```

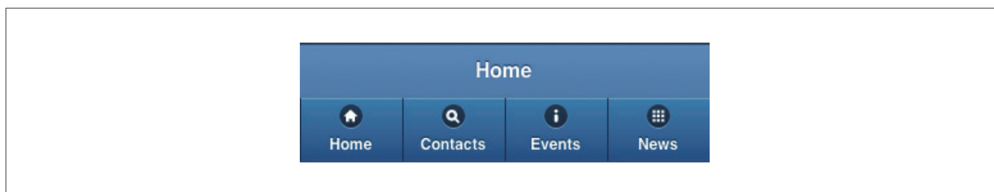


图 3-10：使用图标是可用性与视觉展现的好办法



使用标准的 jQuery Mobile 创建自定义图标的方法，可以在元素上放置任何导航图标。互联网上有大量的免费图标，其中最流行的可以参考 http://glyphish.com。

2. 被选中的元素

每个导航栏都可以有一个被选中的元素，使用 `ui-btn-active` 标识。激活的元素会在当前主题的 UI 基础上高亮（对默认主题来说是亮蓝色）。图 3-11 显示了默认色样的选中样式。

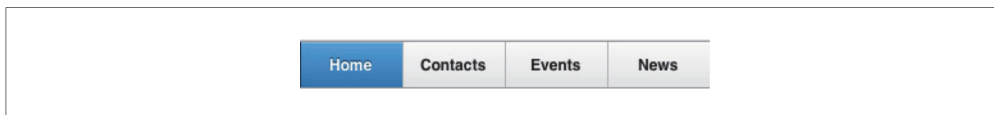


图 3-11：被选中的状态使用一种不同的色样清晰地显示

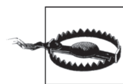
将一个导航栏作为 Web 应用的主导航时，最好将主页作为第一个元素，同时最好在第一个 a 元素上加上 class="ui-btn-active" 以便将它标记为已选中。例如：

```
<div data-role="footer" data-position="fixed">
  <div data-role="navbar">
    <ul>
      <li><a href="#index" class="ui-btn-active">Home</a>
      <li><a href="#contacts">Contacts</a>
      <li><a href="#events">Events</a>
      <li><a href="#news">News</a>
    </ul>
  </div>
</div>
```

用户操作导航元素时，当前选中元素会自动切换为选中状态。也就是说，我们不需要自己更新 ui-btn-active 类。这也意味着，我们可以使用基于 JavaScript 的事件处理程序来管理每个元素的点击事件并改变 UI 以及 / 或者页面导航，同时导航栏的选中状态将会自动更新。

3.1.6　固定页脚

关于导航栏有一点需要理解，如果改变了当前 jQuery Mobile 文档，则当前页脚会被移除，然后显示一个新的页脚。这意味着可能会出现一些渲染问题，比如选中状态（消失又出现，或随着页面过渡出现动画）。要解决这个以及许多其他相关问题，可以使用固定页脚。固定页脚一旦定义，就会固定在 DOM 上，不随页面改变而改变，这样在切换页面时就不再会有任何视觉上的小问题了。

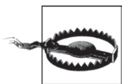


jQuery Mobile 中没有官方支持的定义固定页头的方法。有一些实现方法使用固定页脚以及一些 CSS 技巧来将页脚的位置改到顶部。

要创建固定工具栏，只需在每个需要的页面的固定页脚上定义一个 data-id 标识。然后，任何有相同 data-id 的页脚的页面都将保持前一个页脚而不会更新。

我们还需要在各个页面中使用同样的页脚代码，可能需要更新被选中的项。使用导航栏和固定页脚前进及后退有一个新问题：如何保持选中状态。为了解决这个问题，使用固定页脚上的导航栏时，要在选中项上加两个类：ui-btn-active 和 ui-state-persist。

来看一个完整的例子，其中包含一个作为固定页脚的导航栏，导航栏中有两个页面。结果见图 3-12。



固定页脚应该使用 data-position="fixed" 定义为固定定位。

```
<!DOCTYPE html>
<html>
 <head>
     <!-- 这儿放置典型的 jQuery Mobile 头部 -->
 </head>

 <body>
    <div data-role="page" id="home">

    <div data-role="header">
            <h1>Home</h1>

    </div>

    <div data-role="content">
            <p>This is content for the home</p>
    </div>

    <div data-role="footer" data-id="main" position="fixed">
      <div data-role="navbar">
       <ul>
         <li><a data-icon="home" class="ui-btn-active ui-state-persist">
             Home</a></li>
         <li><a href="#help" data-icon="alert">Help</a></li>
       </ul>
      </div>
     </div>

 </div>

 <div data-role="page" id="help">

    <div data-role="header">
            <h1>Help</h1>
     </div>

    <div data-role="content">
            <p>This is content for Help</p>
     </div>

    <div data-role="footer" data-id="main" position="fixed">
      <div data-role="navbar">
       <ul>
         <li><a href="#home" data-icon="home">Home</a></li>
         <li><a data-icon="alert" class="ui-btn-active ui-state-persist">
             Help</a></li>
       </ul>
      </div>
     </div>
```

```
    </div>

</body>
</html>
```

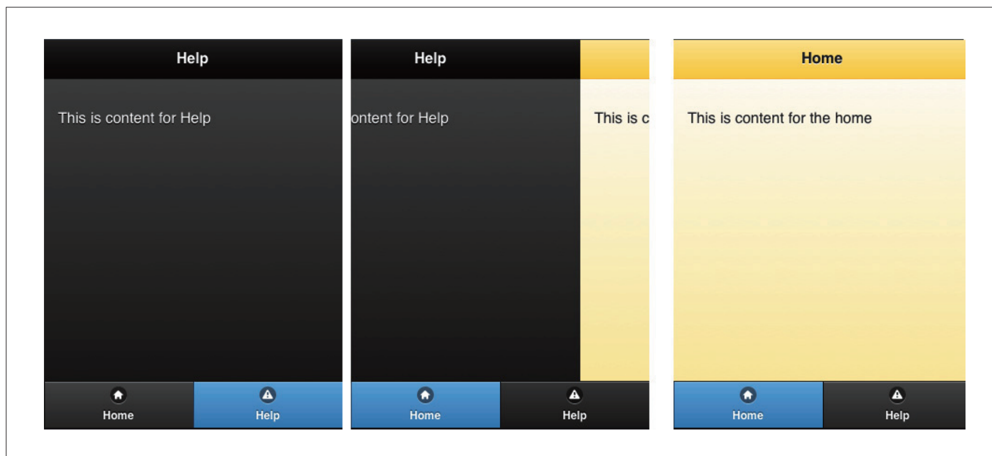


图 3-12：使用固定页脚

3.2 格式化内容

现在，我们看一看 Web 应用的内容区域。最重要的一点：在 `<div data-role="content">` 内任何 HTML 代码都能工作。

jQuery Mobile 的每个主题都包含良好的样式，其中每个标准元素的内边距、外边距、尺寸以及颜色都已针对当前主题及移动设备优化过，被定义过样式的元素包括 hX、链接、粗体及斜体文本、引用、列表以及表格等。



如果想为内容提供自定义的 CSS 格式，需要注意主题系统以避免修改主题时引起 UI 问题。本书后面会有更多关于主题与自定义的内容。

除了基本的 HTML 元素，jQuery Mobile 也提供了一些使用 data-role 定义的组件。就像前面看到的，在页头及页脚处，诸如 hX 或 a 在内的一些元素有自动组件渲染行为。在内容区域，除了一些表单元素外的其他元素都没有这种行为，第 5 章我们会进一步介绍。

图 3-13 展示了一些基本 HTML 元素在 jQuery Mobile 默认主题下的渲染效果。

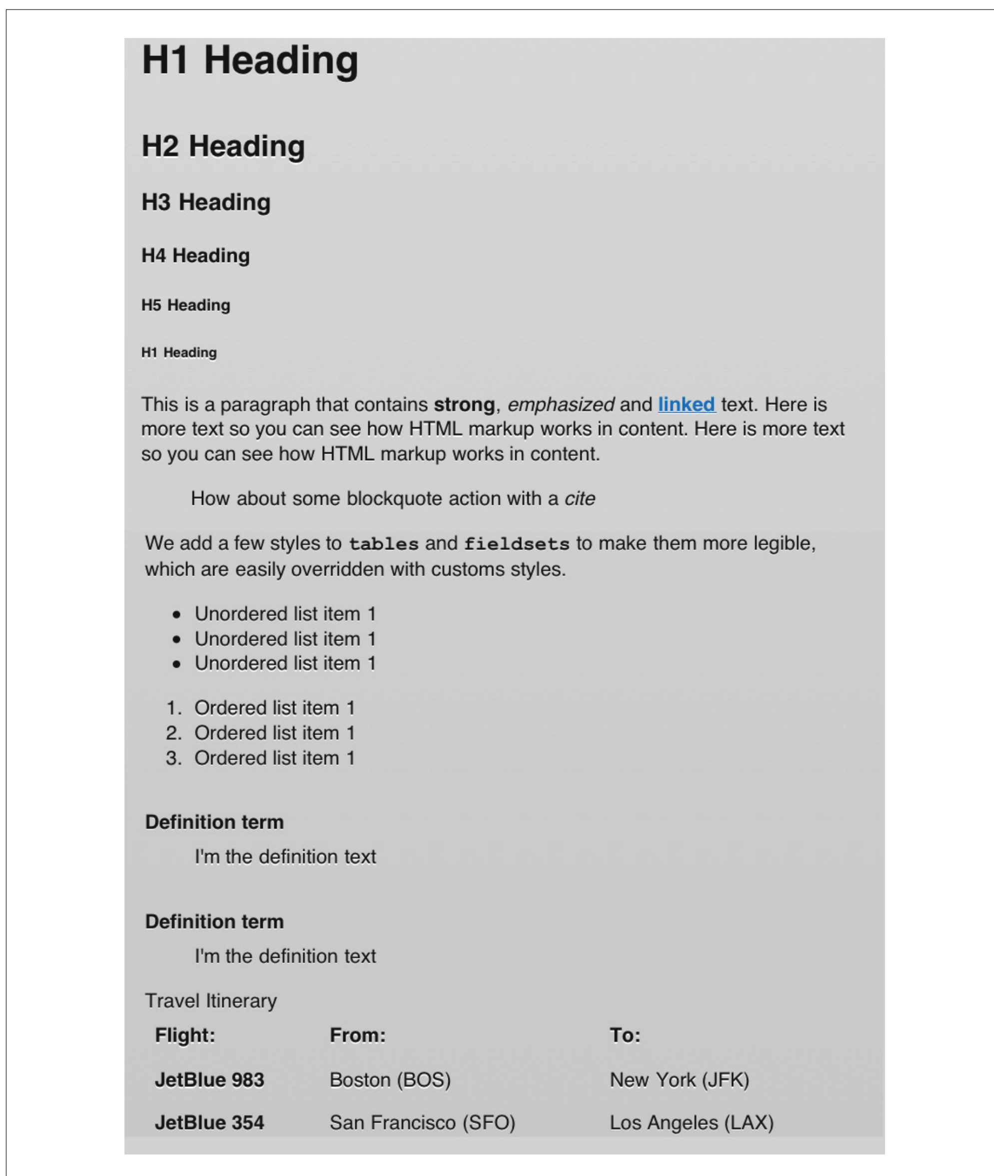


图 3-13：一些基本 HTML 元素在默认主题下的渲染效果

3.2.1 可折叠内容

别忘了我们的目标是移动设备。在移动设备上，空间非常有限，这正是可折叠内容的用武之地。可折叠内容可以通过关联的 JavaScript 行为，在触摸某个标题或按钮时隐藏或者显示。

jQuery Mobile 已经为这种 UI 设计模式提供了支持，不需要我们再编写 JavaScript。要创建可折叠内容，只需定义一个带有 `data-role="collapsible"` 的容器。这个容器需要一个 `hX` 元素作为标题，同时用作开关按钮。可折叠内容则是容器内除了标题外的任何 HTML 代码。

默认情况下，页面加载时 jQuery Mobile 会打开折叠内容。可以在容器上使用 `data-collapsed="true"` 来让它默认收起。

来看一个例子：

```
<div data-role="content">

<div data-role="collapsible">
  <h2>History of Rome</h2>
  <p>There is archaeological evidence of human occupation of the Rome area
     from at least 14,000 years, but the dense layer of much younger debris
     obscures Palaeolithic and Neolithic sites.[11] Evidence of stone tools,
     pottery and stone weapons attest to at least 10,000 years of human
     presence.
  </p>
</div>

<div data-role="collapsible" data-collapsed="true">
  <h2>Government of Rome</h2>
  <p>Rome constitutes one of Italy's 8,101 communes, and is the largest both
     in terms of land area and population. It is governed by a mayor,
     currently Gianni Alemanno, and a city council.
  </p>
</div>

</div>
```

如图 3-14 所示，jQuery Mobile 添加了加号图标和减号图标，分别用作打开和关闭按钮。在写作本书的时候，还不能自定义这些图标。

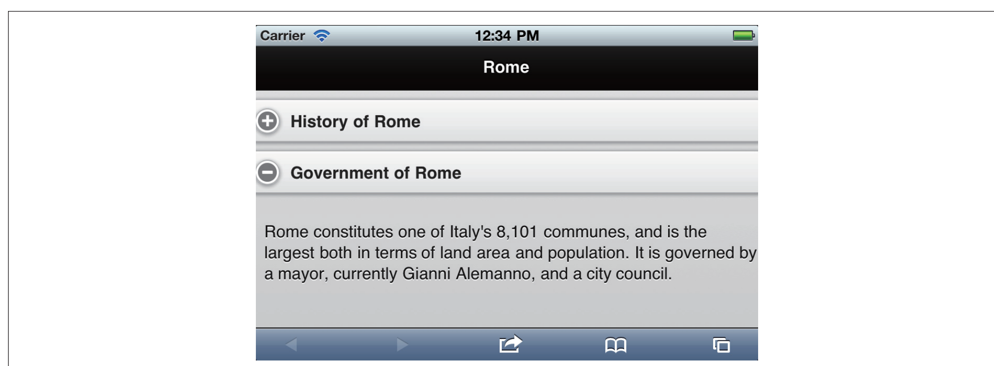
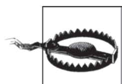


图 3-14：可折叠面板是在同一个页面上展示大量信息的好方法，用户可以自己关闭或打开细节

和其他富控件一样，可以使用 data-theme 来改变这个可折叠面板的色样。还可以指定额外的 data-content-theme 属性，这个属性只影响内容，而不会影响可折叠面板的打开、收起按钮。



如果在可折叠容器中没有定义 hX 元素，则内容将处于打开状态且不能收起。如果定义了多个 hX 元素，则第一个会被用作标题，其他的将被渲染为内部的内容。

嵌套的可折叠内容

可折叠内容面板可以嵌套，jQuery Mobile 会自动在每一级可折叠面板上添加外边距。为避免让 UI 及 DOM 过于复杂，建议不要添加超过两级的嵌套：

```
<div data-role="content">
  <div data-role="collapsible">
     <h2>Rome</h2>
     <div data-role="collapsible">
       <h3>History</h3>
       <p>There is archaeological evidence of human occupation of the
          Rome area from at least 14,000 years, but the dense layer
          of much younger debris obscures Palaeolithic and Neolithic
          sites.[11] Evidence of stone tools, pottery and stone weapons
          attest to at least 10,000 years of human presence. </p>
     </div>
     <div data-role="collapsible" data-collapsed="true">
       <h3>Government</h3>
       <p>Rome constitutes one of Italy's 8,101 communes, and is the
          largest both in terms of land area and population. It is governed
          by a mayor, currently Gianni Alemanno, and a city council. </p>
      </div>
    </div>

    <div data-role="collapsible">
      <h2>Madrid</h2>
      <div data-role="collapsible">
        <h3>History</h3>
        <p>Although the site of modern-day Madrid has been occupied since
           pre-historic times,[23] in the Roman era this territory belonged
           to the diocese of Complutum (present-day Alcalá de Henares).</p>
      </div>
      <div data-role="collapsible" data-collapsed="true">
        <h3>Government</h3>
        <p>The City Council consists of 57 members, one of them being
           the Mayor, currently Alberto Ruiz-Gallardón Jiménez. The Mayor
           presides over the Council.</p>
     </div>
   </div>
</div>
```

3.2.2 手风琴部件

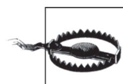
手风琴部件是另一个典型的与可折叠内容相关的富 Internet 应用，它允许将多个可折叠内容聚合起来，这样一次只有一个面板可见，即当你看见某个面板时，所有其他面板都已被收起。

jQuery Mobile 以可折叠集合的形式支持这种部件。也就是一个带有 `data-role="collapsible-set"` 的容器以及一组作为子元素的可折叠面板。

对一个可折叠集合应用 `data-theme` 或 `data-content-theme` 时，除非另有定义，否则所有的子元素都将继承这些值。

模板如下：

```
<div data-role="collapsible-set">
   <div data-role="collapsible">
         <!-- 可折叠标题及内容 -->
   </div>
   <!-- 任意数量的可折叠内容 -->
   <div data-role="collapsible">
         <!-- 可折叠标题及内容 -->
   </div>
</div>
```



默认情况下，jQuery Mobile 会展开可折叠集合中的最后一个面板。如果想默认展开别的面板，只需在想展开的元素上加上 `data-collapsed="false"` 并且在所有其他面板上加上 `data-collapsed="true"`。

来看一个例子：

```
<div data-role="page" id="home">

   <div data-role="header">
        <h1>@firt</h1>
   </div>

   <div data-role="content" data-theme="b">

      <!-- 这儿定义了整个可折叠集合（手风琴）-->
        <div data-role="collapsible-set">
         <div data-role="collapsible" data-collapsed="false">
                <h2>Books</h2>
              <ul>
                 <li>Programming the Mobile Web</li>
                 <li>jQuery Mobile: Up & Running</li>
                 <li>Mobile HTML5</li>
              </ul>
         </div>
```

```
            <div data-role="collapsible" data-collapsed="true">
                   <h2>Talks</h2>
                <ul>
                   <li>Velocity Conference</li>
                   <li>OSCON</li>
                   <li>Mobile World Congress</li>
                   <li>Google DevFest</li>
                </ul>
            </div>
          </div>
         <!-- 可折叠集合（手风琴）结束 -->

      </div>

   </div>
```

图 3-15 显示了一个可折叠集合，其中每个可折叠区域的边框样式都被设置为带圆角。

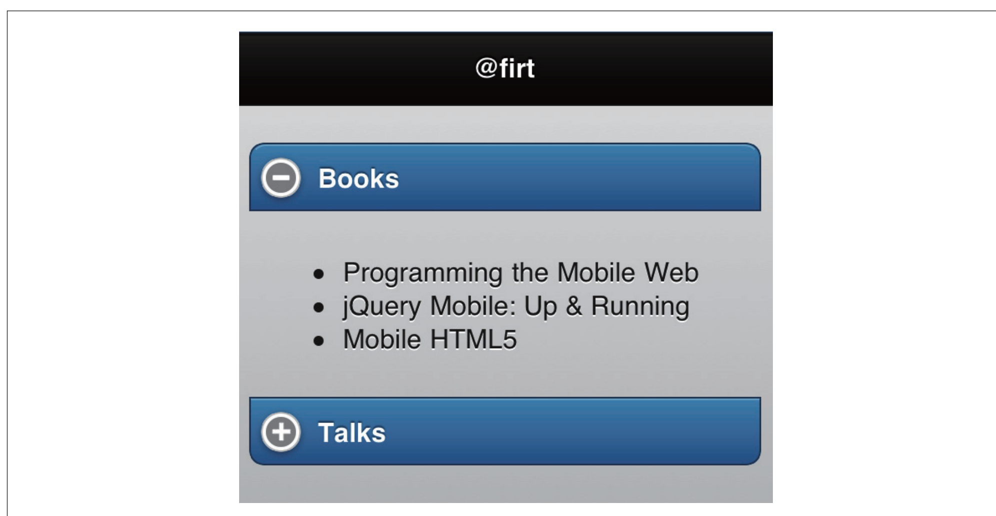
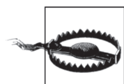


图 3-15：可以使用一个可折叠面板的集合来创建手风琴富控件，一次只有一个面板可见

3.3　列

jQuery Mobile 提供了一些用于将内容按列显示的模板，称为排版网格。这些网格的行为类似表格，但没有使用 table 排版带来的语义问题（除非是表格类型的数据，不然不要使用 table 排版）。



记住，你的应用将运行在移动设备上，使用列应该小心，只放一些小元素，如按钮、链接或小项目。如果目标设备是平板电脑，那列可用的空间则会多一些。

布局网格使用 CSS 类来定义网格区域及列。可以定义 2 至 5 列的网络。网格是不可见的，占满全部 100% 的宽度，没有内外边距。

创建网格只需使用块容器，典型情况为 div，类 ui-grid-a 表示两列，ui-grid-b 表示三列，ui-grid-c 表示四列，ui-grid-d 表示五列。默认情况下，网格的各列宽度相同。

每个单元格都应该是一个带 ui-block-<letter> 的块容器，其中 <letter> 的值为 a 到 e，分别表示网格的第一至第五列。

让我们用 HTML5 中的新元素 section 来尝试一个基本的两列的例子，结果如图 3-16 所示：

```
<div data-role="content">
  <section class="ui-grid-a">
    <div class="ui-block-a">Column 1</div>
    <div class="ui-block-b">Column 2</div>
  </section>
</div>
```

图 3-16：元素最多可以排列为五个等宽列

布局网格的工作方式为平铺排版，也就是说，如果添加的单元格比指定的列数更多，则相当于使用同一个网格模拟不同的行：

```
<div data-role="content">
  <section class="ui-grid-b">
    <!-- 第一行 -->
    <div class="ui-block-a">Cell 1.1</div>
    <div class="ui-block-b">Cell 1.2</div>
    <div class="ui-block-c">Cell 1.3</div>
    <!-- 第二行 -->
    <div class="ui-block-a">Cell 2.1</div>
    <div class="ui-block-b">Cell 2.2</div>
    <div class="ui-block-c">Cell 2.3</div>
  </section>
</div>
```

3.4 按钮

我们已经看到，a 元素可被用于创建页面之间及指向外部内容的链接。不过，典型的 a 元素的渲染效果并不适合触摸设备，a 元素通常是内联的，只有文字是可点击区域，这对使用触摸的用户来说并不是好的体验，因此，jQuery Mobile 提供了按钮。

按钮是一种 UI 部件，感觉就像……，好吧，就像按钮。它是一个带文本的更大的可点击区域，可能还有图标。

创建按钮的方式有好几种：

- 使用 button 元素；
- 使用会被渲染为按钮的 input 元素，包括 type="button"、type="submit"、type="reset" 以及 type="image"；
- 任何带有 data-role="button" 的 a 元素。

jQuery Mobile 的按钮一般的渲染效果为带有居中的文字、圆角及阴影，取决于浏览器对 CSS3 的支持情况。



显示一组按钮时，一个良好的 UI 设计模式是让那个肯定按钮或者最有可能被点击的按钮使用与众不同的主题。

默认情况下按钮会占满屏幕的整个宽度，即每个按钮会独占一行。下面是一个典型的例子，效果见图 3-17。

```
<a href="#" data-role="button">Click me!</a>
<button data-theme="b">Click me too!</button>
<input type="button" value="Don't forget about me!">
```

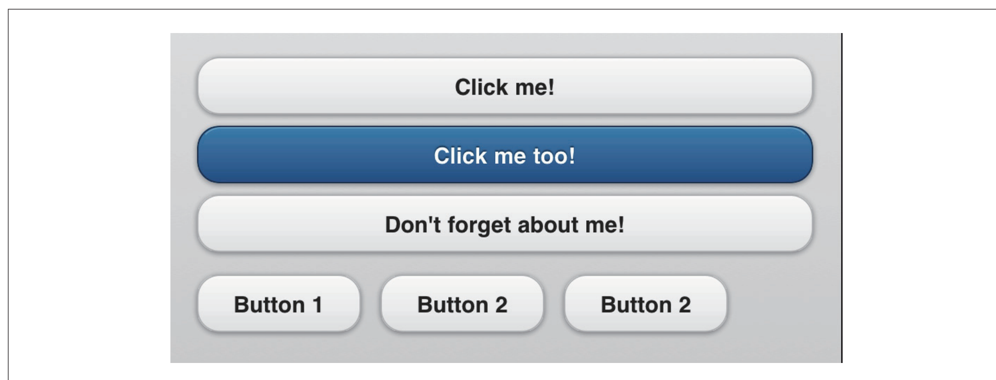


图 3-17：在 Web 应用中，按钮是为触摸设备提供交互的最好方法

3.4.1 内联按钮

可以在元素上应用属性 `data-inline="true"` 来创建内联按钮，这种按钮不会占满屏幕宽度。这样，用下面的代码可以在同一行放置三个按钮（如图 3-17 所示）。

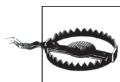
```
<a href="#" data-role="button" data-inline="true">Button 1</a>
<a href="#" data-role="button" data-inline="true">Button 2</a>
<a href="#" data-role="button" data-inline="true">Button 2</a>
```



如果想用内联按钮来占满整个屏幕宽度，可以使用布局网格来定义至多五个列，每列包含一个按钮。

3.4.2 分组按钮

如果有若干个按钮彼此相关，可以将它们组合起来，这样会得到一个不同的用户界面，其中每个按钮都位于一个分组容器中。在 jQuery Mobile 中这种技术被称为分组控件，它包含一个叫作控件组的新部件以及一组按钮。



使用分组按钮时，不要将它们声明为内联按钮。

控件组只是一个容器，一般为一个带 `data-role="controlgroup"` 的 `div` 元素。下面这个分组将包含一组按钮，看起来与图 3-17 类似：

```
<div data-role="controlgroup">
  <a href="#" data-role="button">Button 1</a>
  <a href="#" data-role="button">Button 2</a>
  <a href="#" data-role="button">Button 2</a>
</div>
```

如图 3-18 所示，按钮在垂直方向上彼此相连。也可以通过在控件组上使用 `data-type="horizontal"` 属性来创建一个水平布局，如图 3-19 所示：

```
<div data-role="controlgroup" data-type="horizontal">
  <a href="#" data-role="button" data-inline="true">Button 1</a>
  <a href="#" data-role="button" data-inline="true">Button 2</a>
  <a href="#" data-role="button" data-inline="true">Button 2</a>
</div>
```



图 3-18：有多个内联按钮时，最好将它们包装到控件组中以便获得更好的外观



图 3-19：如果按钮数少于 5，并且文案较短，可以使用水平控件组



水平按钮组看起来可能像一个不同选项的选择器（可按下按钮）。不过，到目前为止，它们还只是组合到一起的按钮，使用这个技术时它们并没有被选中状态。本书后面的章节将涉及处于按下状态的按钮。

3.4.3　效果

默认情况下，每个按钮都被渲染为带圆角和阴影。可以通过布尔值属性 data-cornters 和 data-shadow 来改变这个行为。本书后面的章节会讲解如何使用 JavaScript 来改变这两个属性的全局定义。

下面的代码创建了一个不带圆角和边框的按钮：

```
<a href="#" data-role="button" data-shadow="false" data-corners="false">
Help</a>
```

3.4.4　图标

jQuery Mobile 包含一个强大的图标机制，允许我们对按钮应用主题，就像对其他可设置图标的部件所做的一样。因此，本书后面的章节会再次讨论（其他部件的）图标，它们的选项、图标库以及功能与我们现在正在讨论的完全一致。

每个按钮都可以通过 data-icon 属性定义一个图标。这个属性接收一个图标名，名字可以是图标库（框架已经包含的图片）中已有的某个图片名，也可以自定义。

jQuery Mobile 1.0 中可用的图标见表 3-1。

表3-1：jQuery Mobile中可用的图标

图标描述	值	图标描述	值
向左箭头	`arrow-l`	刷新	`refresh`
向右箭头	`arrow-r`	向前动作	`forward`
向上箭头	`arrow-u`	向后动作	`back`
向下箭头	`arrow-d`	网格	`grid`
删除（*x*）	`delete`	星星	`star`
加号	`plus`	警报（警号图标）	`alert`
减号	`minus`	信息（i）	`info`
检查标记	`check`	主页图标	`home`
齿轮图标（设置）	`gear`	搜索图标	`search`



记住，这些图标图片由框架提供，位于下载的 ZIP 包中或托管在 CDN 上。如果决定自己托管框架文件，或者正在创建一个嵌入了 jQuery Mobile 的原生应用，则需要确保对应的图片文件夹可用。jQuery Mobile 使用 CSS Sprites 技术来减少图标对应的图片加载，图标有低分辨率版和高分辨率两个版本，框架会自动决定使用哪个版本。

图 3-20 显示了每个图标的渲染效果。

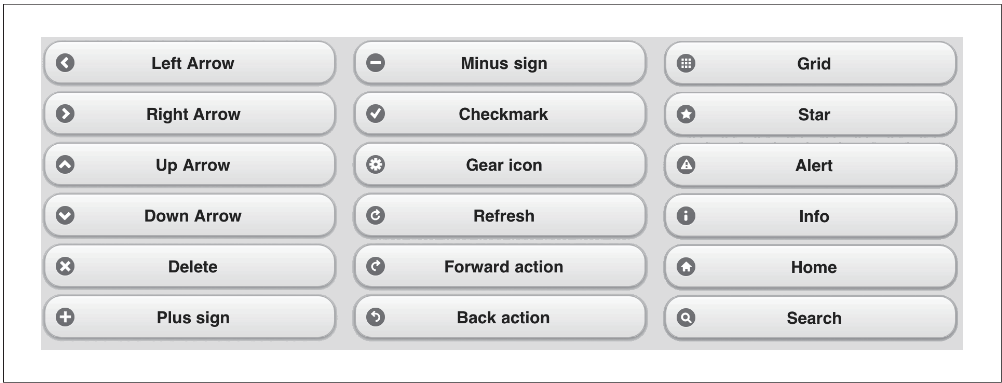


图 3-20：jQuery Mobile 框架中包含的所有图标

3.4.5 创建自定义图标

想使用自定义图标，只需在 `data-icon` 属性中定义一个不重复的名字。推荐的命名机制为 < 应用名 >-< 图标名 >，例如：`myapp-tweet`。



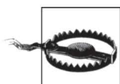
jQuery Mobile 会自动为图标添加白色圆形背景，所以无论行背景是什么都不会有问题。

jQuery Mobile 还需要我们创建一个名为 `ui-icon-<name>` 的 CSS 类，例如 `ui-icon-myapp-tweet`，其中指定一个背景图片。为了减少 UI 问题，图标应该是白色（或透明）背景，18×18 像素，带 alpha 透明的 PNG-8 格式。图标不应该带边框，使用列表视图时框架会自动添加边框。图标还应该比 18×18 像素小一些，因为它将被渲染在一个 18 像素宽的圆形中。同时，由于图标会被绘制在圆形中，所以不要使用边角。

对应的 CSS 类应该类似下面的代码：

```
<style>
   .ui-icon-myapp-tweet {
       background-image: url(icons/tweet.png);
   }
</style>
```

我们的自定义图标在 iPhone 4、iPod Touch 4G 等使用视网膜显示屏的高分辨率设备上无法工作，因为还需要一个该图标的高分辨率版本。要实现这一点，需提供一个备选的 CSS 选择定义。图片则应该扩大一倍至 36×36 像素。为低分辨率及高分辨率提供同一张图片也是可以的，不过，为了更好的显示效果，推荐使用两张图片。



如果某些 CSS 样式只会应用在一个 HTML5 jQuery Mobile 文档上，最好使用 `style` 标签将它们加到文档中，而不是引用外部资源，以便提高性能。当然，将它们与其他 CSS 写在一起会更简单，但考虑一下添加新的外部资源时的性能吧，除非你有一个活跃的缓存策略。

要提供两个版本，只需使用 CSS3 媒体查询格式写两个不同的定义。第一个定义将应用到所有分辨率，第二个则将应用到两倍于标准分辨率的设备（如 iPhone 4 或更新）上。需要强制背景尺寸为 18×18，因为在这些设备上 px 是虚拟的点，每一个对应两个真实的像素。对应的 CSS 类现在看起来类似这样：

```
<style>
   .ui-icon-myapp-tweet {
       background-image: url(icons/tweet.png);
   }

   @media only screen and (-webkit-min-device-pixel-ratio: 2) {
      .ui-icon-myapp-tweet {
          background-image: url(icons-hd/tweet.png) !important;
            background-size: 18px 18px;
      }
}

</style>
```

让我们看看标准按钮与自定义按钮的对比，见图 3-21。自定义按钮使用了我们刚刚定义的 CSS 样式：

```
<a href="#" data-role="button" data-icon="gear">Help</a>
<a href="#" data-role="button" data-icon="myapp-tweet">Tweet</a>
```

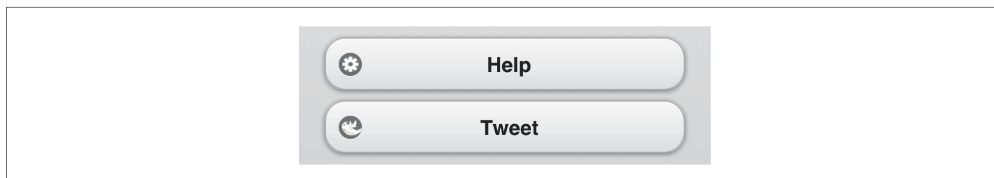


图 3-21：可以为按钮及其他 UI 部件提供自定义图标

3.4.6 图标位置

默认情况下，每个图标都被渲染在按钮文字的左侧。可以使用 data-iconpos 属性来改变这个位置，可用的值有 right、left（默认值）、bottom 以及 top。使用 bottom 或 top 时，按钮的高度会增加，在导航栏或水平分组按钮中这两个属性特别有用。

下面来看同一链接的不同图标位置（参见图 3-22）：

```
<a href="#" data-role="button" data-icon="help" data-iconpos="right">Help
    </a>
<a href="#" data-role="button" data-icon="help" data-iconpos="left">Help</a>

<div data-role="controlgroup">
  <a href="#" data-role="button" data-icon="help" data-iconpos="bottom">
      Help</a>
  <a href="#" data-role="button" data-icon="help" data-iconpos="top">Help </a>
      </div>
```

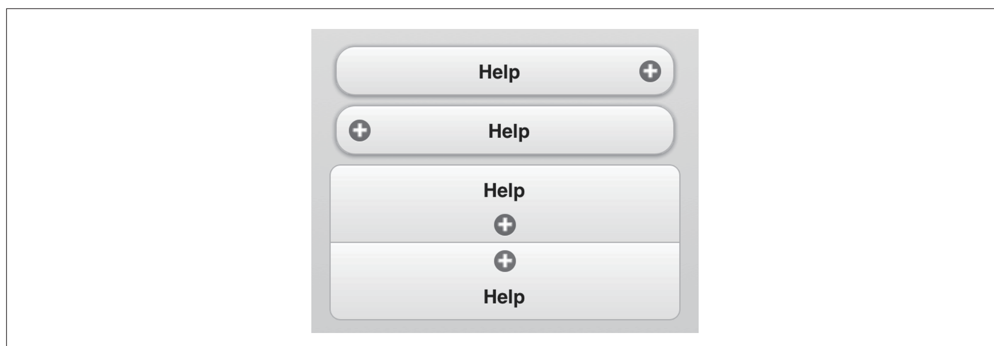


图 3-22：使用 **data-iconpos** 可以将图标放置在四个方向的任何一处

3.4.7 纯图标按钮

jQuery Mobile 也支持没有文字的按钮。由于这种按钮对触摸界面来说太小了，因此这个特性并不常用。要创建一个纯图标按钮，只需在按钮上指定 `data-iconpos="notext"`。

3.4.8 图标阴影

jQuery Mobile 包含一个可移除图标阴影效果的属性。要实现这个效果，只需在按钮元素上指定 `data-iconshadow="false"`。

第4章
列表

到现在为止，我们已经使用 jQuery Mobile 创建了若干非常简单的页面。下一步是学习使用框架提供的各种控件和视图。本章将讨论列表。几乎所有移动项目的内容中都至少会包含一个列表，所以我们首先关注列表。

要从列表的定义开始说起有点儿难，大家都知道列表是什么，作为一个通用术语，这儿也没有新东西可介绍。在 jQuery Mobile 的世界里，列表就是页面中的一个有序（`ol` 元素）或无序（`ul` 元素）列表，其中至少包含一个列表项（`li` 元素），并且使用 HTML5 语法 `data-role="listview"` 将对应的 `role` 的值定义为 `listview`。

如果只想显示普通的无序或有序列表，只需使用标准的 `ul` 或 `ol` HTML 元素即可，记住不要给它们添加任何 jQuery Mobile `role` 属性。

在下一章我们将看到，列表在 jQuery Mobile 框架中是一个强大并且用途非常广泛的解决方案。

最典型的列表是无序列表（`ul`），并且页面中除了这个列表外没有别的内容。来看一个例子，结果见图 4-1：

```
<!DOCTYPE html>
<html>

<head>
<meta charset="utf-8" />
 <title>Up and Running with jQuery Mobile</title>
 <link rel="stylesheet" href="jquery.mobile-1.0.min.css" />
```

```
 <script src="jquery-1.6.4.min.js"></script>
 <script type="text/javascript" src="jquery.mobile-1.0.min.js">
 </script>
 <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body>

 <div data-role="page" id="main">
     <div data-role="header">
        <h1>HTML5 and APIs</h1>
     </div>

     <div data-role="content">
        <ul data-role="listview">
            <li>Offline Access
            <li>Geolocation API
            <li>Canvas
            <li>Offline Storage
            <li>New semantic tags
        </ul>
     </div>

  </div>

 </body>
</html>
```

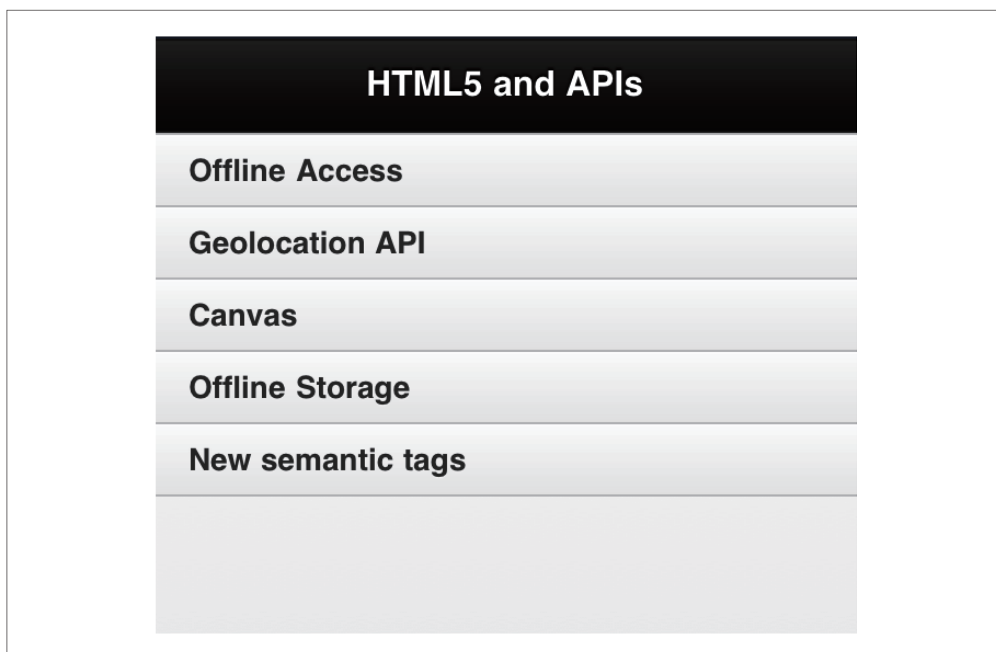


图 4-1：典型的列表视图，各行的渲染效果很适合触摸设备

从 jQuery Mobile 框架默认使用的行高可以发现，列表的渲染已为触摸操作优化过。每个列表项都自动占满整页宽度，这是典型的触屏设备 UI 模式。（对长列表来说，可以使用固定工具栏。）



记住，jQuery Mobile 是在客户端工作的。带列表的文档可以由任何服务端平台，如 PHP、Java、ASP.NET 或 Ruby 等动态生成。

也可以使用 `ol` 元素来创建有序列表。不过，如果没有包含带链接的交互行的话，它的外观与使用 `ul` 将没有区别，如图 4-2 所示：

```
<!DOCTYPE html>
<html>

<head>
  <!-- 这儿放置典型的 jQuery Mobile 头部 -->
</head>

<body>

 <div data-role="page" id="main">

   <div data-role="header">
     <h1>Chapters</h1>
   </div>

   <div data-role="content">
      <ol data-role="listview">
          <li>The Mobile Jungle
          <li>Mobile Browsing
          <li>Architecture and Design
          <li>Setting up your environment
          <li>Markups and Standards
          <li>Coding Markup
          <li>CSS for Mobile Browsers
          <li>JavaScript Mobile
          <li>Ajax, RIA and HTML5
          <li>Server-Side Browser Detection
          <li>Geolocation and Maps
          <li>Widgets and Offline webapps
          <li>Testing, Debugging and Performance
          <li>Distribution and Social Web 2.0
      </ol>
   </div>

  </div>

 </body>
</html>
```

这个例子显示了 jQuery Mobile 中列表行的渲染效果。如果文本超出了一行，框架

将自动调整该行的尺寸。

Chapters

The Mobile Jungle

Mobile Browsing

Architecture and Design

Setting up your environment

Markups and Standards

Coding Markup

CSS for Mobile Browsers

JavaScript Mobile

Ajax, RIA and HTML5

图 4-2：对非交互列表而言，有序列表与无序列表的外观一样



HTML5 不使用 XML 语法，因此无需关闭所有标签，如 li 元素。当然，如果你不习惯，也可以关闭它们（如果你和我一样觉得没关闭的 li 很刺眼的话）。

4.1 整页列表与插入列表

默认情况下列表使用整页模式，也就是说列表将覆盖整个页面内容，如图 4-1 及图 4-2 所示。不过，也有些项目会用到与其他 HTML 内容混合的列表。为了处理这种情况，jQuery Mobile 提供了内联列表。要定义这种列表，只需在对应的 ul 或 ol 元素上添加 data-inset="true" 即可：

```
<ol data-role="listview" data-inset="true">
    <!-- 行元素 -->
</ol>
```

在图 4-3 中可以看到之前的例子使用内联模式渲染的效果。从图中可以看到，表格

的外观与之前有些不同，与页面中其他内容之间的边距更大，同时还添加了圆角等 CSS3 特效。

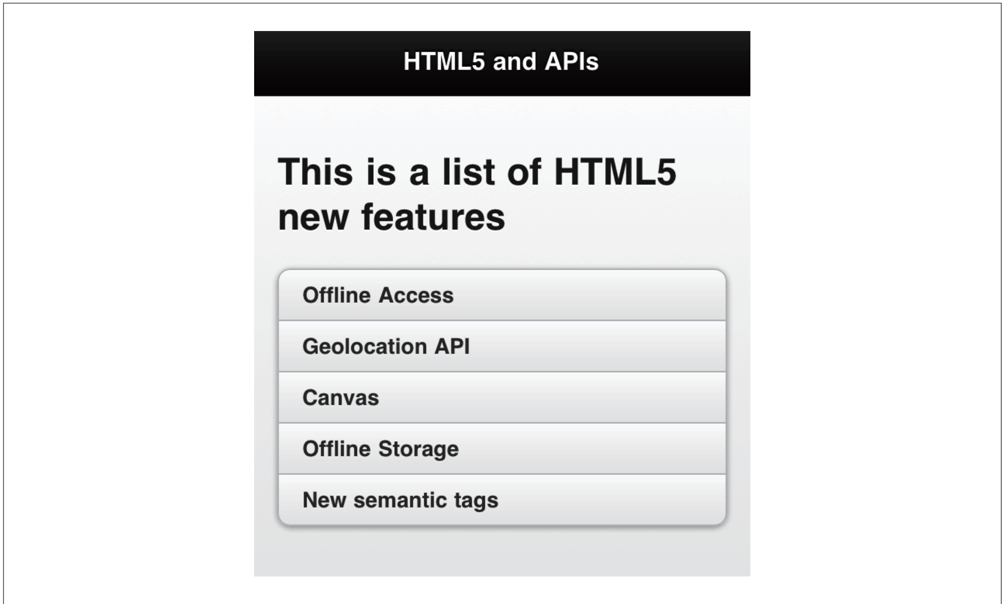


图 4-3：内联表格有着不同的外观，可以与其他内容共享一个页面，包括其他列表

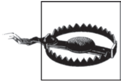


可以通过 `ul` 元素上的 `data-theme` 属性指定整个列表的色样，也可以为每个 `li` 元素单独指定色样。

用内联列表可以设计出包含多个表格的页面，表格之间还可以插入其他内容。

4.2 视觉分隔符

分隔符用于将一个列表划分为两个各自带标题的部分，如图 4-4 所示。在 iPhone 及 iPad 等 iOS 移动操作系统上，这是一个常用的设计模式。要在 jQuery Mobile 中实现这个功能，只需在 `li` 元素上使用新的 `role` 属性：`data-role="list-divider"`。



注意列表分隔符也是标准的 `li` 元素，只是 `role` 不同，它们和普通行在 DOM 树中是同级的。

可以使用这个技术将列表元素划分为不同的组，例如在展示以字母排序的数据时可以用 A-Z 的字母（或者其他任何分类方法）分组。可以通过改变色样的方式高亮每个分隔符。

来看一个简单的例子：

```
<!DOCTYPE html>
<html>

<head>
  <!-- 这儿放置典型的jQuery Mobile头部 -->
</head>

<body>

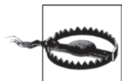
 <div data-role="page" id="main">

   <div data-role="header">
     <h1>World Cup</h1>
   </div>

   <div data-role="content">
      <ul data-role="listview">
          <li data-role="list-divider">Group A
          <li>Argentina
          <li>Nigeria
          <li>England
          <li>Japan
          <li data-role="list-divider">Group B
          <li>United States
          <li>Mexico
          <li>Korea
          <li>Greece
          <li data-role="list-divider">Group C
          <li>Germany
          <li>Finland
          <li>Chile
          <li>South Africa
      </ul>
   </div>

  </div>

 </body>
</html>
```



如果列表是用 JavaScript 等客户端代码生成的，在生成之后必须调用一个刷新方法，通知框架应用改变并正确渲染列表。后面我们会谈到这些技术，现在只需要知道，如果页面上只有一个列表，页面 `id` 为 `page1`，则对应的 jQuery 代码将形如 `$("#page1 ul").listview("refresh")`。

图 4-4 展示了列表分隔符的效果。

World Cup

Group A

Argentina

Nigeria

England

Japan

Group B

United States

Mexico

Korea

Greece

图 4-4：上图展示了行分隔符在列表中的渲染效果

4.3 交互行

列表具备了触摸交互功能后将更加强大。

如果一个列表元素包含 a 元素，则该行会被转换为一个可响应触摸及 / 或鼠标导航的交互行。jQuery Mobile 有一个很好的特性，只要某行中包含链接，无论这个链接在行中的什么位置，该行整行都会响应触摸事件，也就是说用户不用非得点击在链接文本的区域，只需点在行内的任何一个位置即可。



交互行的高度与只读行不同，这是为触摸交互优化过的设计，因为触摸区域需要足够大以避免误点。交互行中的文本也会更大一些。在基于 iOS 的设备上，大多数控件都建议使用 44 像素的尺寸。其他设备则使用小一些的尺寸。

jQuery Mobile 会自动在交互行的右侧添加漂亮的箭头，提醒用户该行是可触摸的，如图 4-5 所示。可以用 `data-icon` 来改变这个图标，见下面的示例。

同一个列表中可以混合使用交互行和只读行。不过，列表的典型 UI 设计是要么全是交互行，要么全是只读行。

下面是一个交互行的实例，见图 4-5：

```
<!DOCTYPE html>
<html>

<head>
  <!-- 这儿放置典型的 jQuery Mobile 头部 -->
</head>

<body>

 <div data-role="page" id="main">

   <div data-role="header">
     <h1>Interactive</h1>
   </div>

   <div data-role="content">
      <ul data-role="listview">
          <li><a href="#page2">Internal Page link</a>
          <li><a href="other.html">External Page link</a>
          <li><a href="http://www.mobilexweb.com">Absolute external link</a>

          <li><a href="tel:+13051010200">Call to a phone number using a link</a>
          <li><a href="javascript:alert('Hi!')">JavaScript link</a>
      </ul>
   </div>

  </div>

 </body>
</html>
```

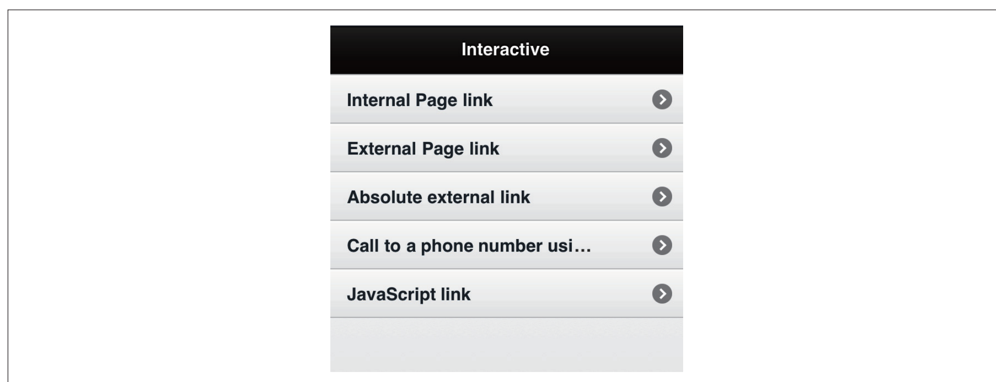


图 4-5：交互行只是包含超链接的列表元素，它们将自动被转换为可响应点击或触摸事件的区域

如图 4-5 所示，使用交互行时，jQuery Mobile 会尝试保持各行统一行高。因此，如上面第四行的标题所示，当某行的标题超过了一行时 jQuery Mobile 会裁剪该行文本，并在支持 CSS3 的设备上在文本最后添加省略号。如果没有显示完整文本的详情页面的话，最好让各行标题的文本尽量简短。

a 元素必须在 li 标签内，不需要插入行标题作为链接内容。空链接也可以工作：

```
<li><a href="#page2"> Internal Page link </a>
```

在一些支持鼠标或焦点导航的设备上，如 BlackBerry 或 Android，用户可能会使用键盘或轨迹球（trackball）进行导航。激活一行（通过某个 OK 键或按下轨迹球）时该列表元素内的链接将被激活。从图 4-6 中可以看到，在 Android 设备上，浏览器为选中行添加了一个通常为橙色的焦点边框。

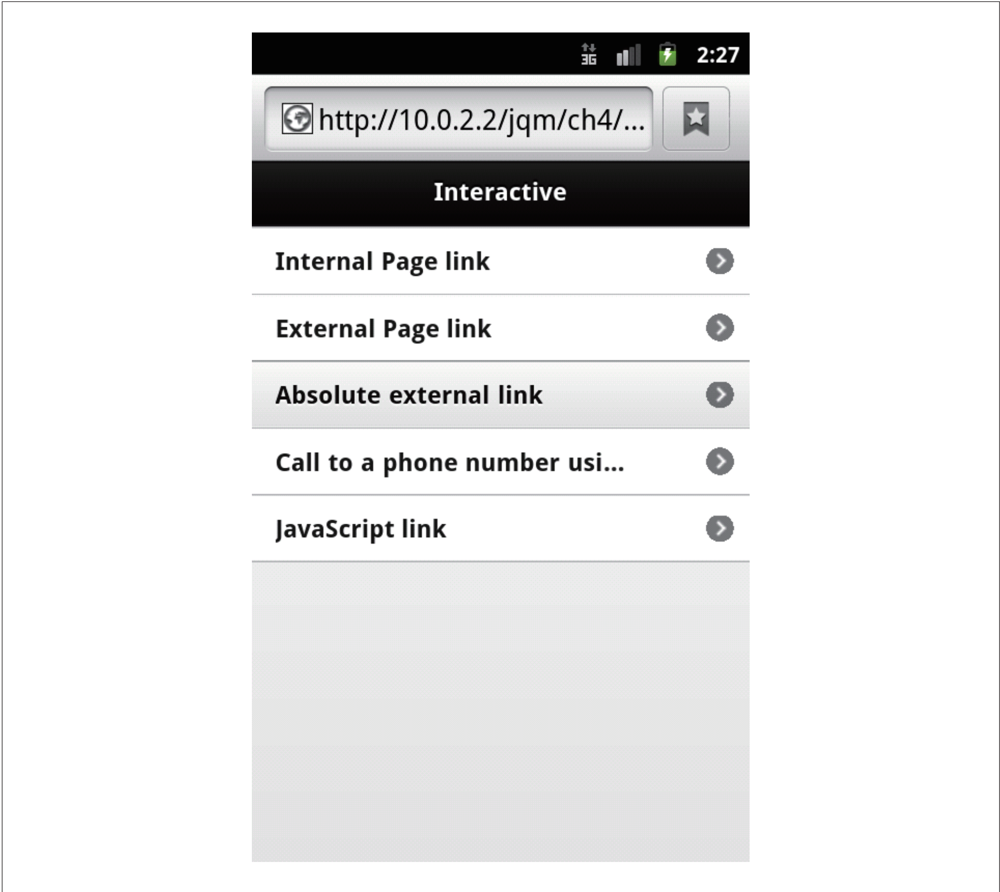


图 4-6：在基于焦点的浏览器上，可以通过鼠标键或轨迹球来浏览交互列表

推荐在页面中使用交互列表来创建链接，而不是单用 a 标签，因为前者专门为触摸及基于鼠标的导航优化过。

用户点击一个交互行时，jQuery Mobile 内会产生一个页面转向动作，在转向或从网络下载新页面期间被点击的行会处于选中状态，并使用不同的色样，如图 4-7 所示。

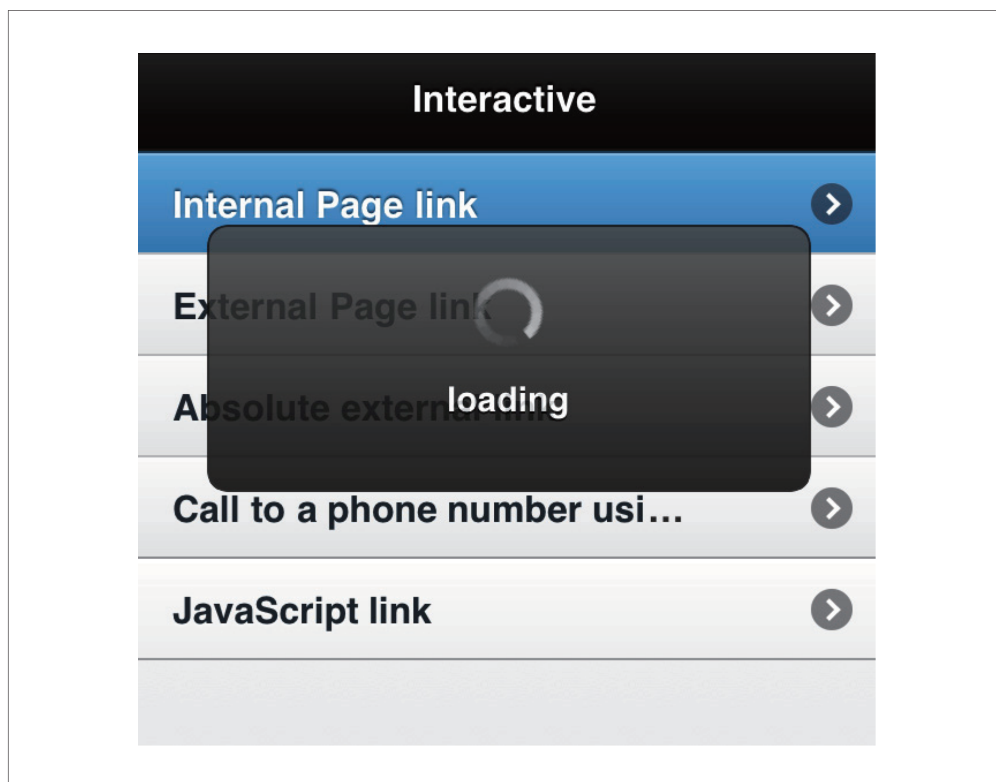


图 4-7：选中一个交互行时，框架会将该行的色样改为高亮背景，在默认主题中是亮蓝色

可以使用 `li` 元素上的 `data-icon` 属性来改变交互行的默认图标，例如：

```
<li data-icon="info"><a href="#moreinfo">More information</a></li>
```

也可以通过指定 `data-icon` 的值为特殊值 `false` 来移除交互行的图标：

```
<li data-icon="false"><a href="#page2">No icon interactive row</a></li>
```

4.3.1　内嵌列表

内嵌列表是 jQuery Mobile 的一个很棒的特性。如果想在不同的页面显示继承结构或导航，比如“洲➤国家➤城市”，则可以使用内嵌列表。

内嵌列表只是位于一个列表视图（一个指定 role 的列表）某项中的另一个列表视图。例如，可以在某个 ul 中的 li 内再包含一个 ul。jQuery Mobile 会隐式地为每个内嵌列表生成一个页面，如同我们显式生成的一样，并会自动处理各个层级的联系。

内嵌列表会使用不同的渲染主题，以便用户能在视觉上识别出它是一个二级列表。当然，也可以使用后面会讲到的 data-theme 来显式地指定主题。

于是，在一个页面上就能有不同层级的导航。页面加载时，jQuery Mobile 只会显示第一级的列表项，每个包含内嵌列表的列表项都会成为交互行。用户选择这样的行时会转向显示下一级列表视图的新页面，对应的后退操作将返回上一个层级。

内嵌列表的层级数量没有限制，不过，如果层级及列表项很多，最好使用不同的页面，以便减少 DOM 和初始化加载的时间。推荐只在特定情形下使用内嵌列表，可不要把整个站点都放在一个内嵌列表里。

当然，内嵌列表和普通交互行可以混合使用。因此，可以创建这样的列表，其中一些列表项指向其他文档或页面，另一些列表项则包含内嵌列表。

除了 ul 或 ol 之外，记得在对应的 li 中添加额外的文本，因为框架需要为隐式创建的页面中的交互项显示一个标题。

因此，一个典型的内嵌列表看起来像这样：

```
<li>
    Item title
    <ul data-role="listview">
        <!-- 这儿放内嵌列表项 -->
    </ul>
```

让我们创建一个示例，结果如图 4-8 所示：

```
<!DOCTYPE html>
<html>

<head>
  <!-- 这儿放置典型的 jQuery Mobile 头部 -->
</head>

<body>

 <div data-role="page" id="main">

   <div data-role="header">
     <h1>Training</h1>
   </div>

   <div data-role="content">
```

```
        <ul data-role="listview">
            <li><a href="order.html">Order Now!</a>
              <li>Cities available
                    <ul data-role="listview">
                          <li>Boston
                          <li>New York
                          <li>Miami
                          <li>San Francisco
                          <li>San Jose
                     </ul>

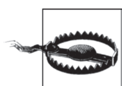
              <li>Topics
                    <ul data-role="listview">
                          <li>Intro to Mobile Web
                                <ul data-role="listview">
                                    <li>WML and other oldies
                                    <li>XHTML MP
                                    <li>HTML 4.01
                                    <li>HTML5
                                </ul>
                          <li>Mobile Browsers
                                <ul data-role="listview">
                                    <li>Safari for iOS
                                    <li>Android Browser
                                    <li>Firefox for Mobile
                                    <li>Opera
                                </ul>
                          <li>Tablet Browsers
                          <li>Using jQuery
                    </ul>

              <li>Promotions
                    <ul data-role="listview">
                          <li>10% off before May
                          <li><a href="promo3x2.html">3x2</a>
                          <li>10% off to subscribers
                    </ul>

        </ul>
     </div>

    </div>

   </body>
  </html>
```



在内嵌表中，jQuery Mobile 会自动使用行中的文本作为新创建的页面的标题，因此需要让这些文本尽量简短，以免超出标题区域。

在图 4-8 中可以看到使用无特效的内嵌列表生成的所有导航及页面。

图 4-8：内嵌列表的导航看起来像有不同的页面，但实际上并没有创建新页面



通过一些 jQuery Mobile 技巧，可以创建能自动更改 URL 散列值的链接，指向内嵌列表自动生成的页面。不过，如果真需要用到这个功能的话，恐怕最好还是显式地创建外部页面，而不要用内嵌列表。

4.3.2　分割按钮列表

有时，我们需要在同一行包含两个交互操作。最常见的场景是一个详情操作和一个编辑操作。

例如，把联系人姓名在表格中列出时，可以提供两个可能的操作：查看详情和编辑。为了实现这一点，jQuery Mobile 使用了分割行控件。如果一个列表项 `li` 包含两个超链接（a 元素），则它将被自动当作分割行处理。

如图 4-9 所示，各行被分割成了左右两部分。DOM 中的第一个链接会被用作第一个操作，即该行左侧的操作。第二个链接会被用作另一个操作，即该行右侧的操作。

第二个链接操作的链接中可以不包含文本，事实上第一个链接操作中也可以没有文本。如果让 a 链接内容为空，那么它将被应用到整行上。



根据 iOS 用户界面准则，第一个操作应该是详情，第二个操作应该是编辑。不过这并不是强制规定。

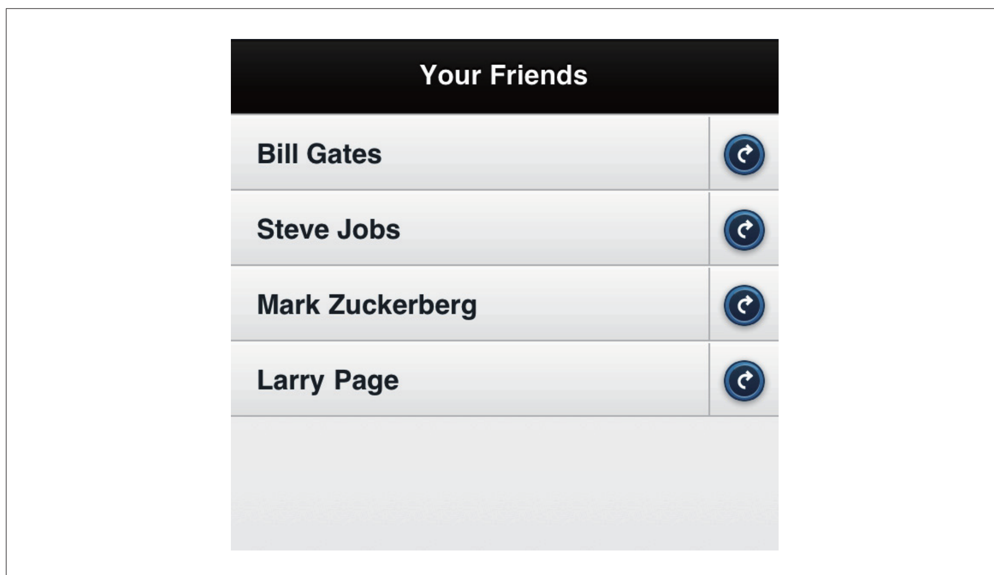


图 4-9：分割行允许我们在一行中包含两个操作，一个用于典型操作，另一个带特殊图标的位于右侧

默认情况下，jQuery Mobile 会为右侧的按钮（第二个操作）使用一个带边框的箭头图标，与标准交互行稍有不同。

可以在对应的 `ul` 标签上通过 `data-split-theme` 来为右边的图标定义不同的主题。

看一个例子，结果如图 4-10 所示：

```
<!DOCTYPE html>
<html>

<head>
  <!-- 这儿放置典型的 jQuery Mobile 头部 -->
</head>

<body>

 <div data-role="page" id="main">

   <div data-role="header">
     <h1>Your Friends</h1>
   </div>

    <div data-role="content">
       <ul data-role="listview">
           <li><a href="details/bill">Bill Gates</a>
               <a href="edit/bill"></a>
```

```
                <li><a href="details/steve">Steve Jobs</a>
                    <a href="edit/steve"></a>

                <li><a href="details/mark">Mark Zuckerberg</a>
                    <a href="edit/mark"></a>

                <li><a href="details/larry">Larry Page</a>
                    <a href="edit/larry"></a>
           </ul>
        </div>

       </div>


      </body>
    </html>
```

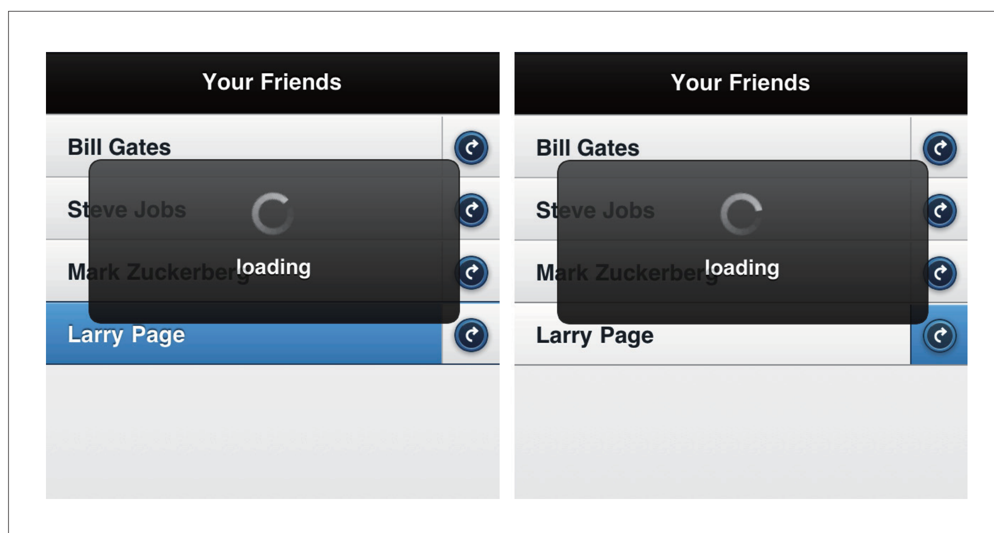


图 4-10：如图所示，分割行的每个区域都有自己的选中状态

可以使用 `data-split-icon` 来修改第二个操作的图标，这个属性需应用在列表自身上（`ul` 或 `ol` 元素），而不是列表项上（`li` 元素）。

列表视图与按钮（下一章会讲到）使用同一组图标，不过带圆角边框，从而与普通按钮有所区别。

jQuery Mobile 1.0 中可用的图标见表 3-1。



可以在一张图片上提供自定义的图标集，再通过 CSS Sprites 技术使用，就像 jQuery Mobile 框架中所用的图片一样。

控制行的重要程度

如果想让某些行或所有行显得比普通行更重要，可将对应行的标题包含在 hx 标签中，如 h2 或 h3，这样这些行的行高将会增加一些。

如果想让某些行显得不那么重要（比如那些不常用的操作），可以将对应行的标题包含在 p 标签中，这样这些行的行高将会减少一些。

在后面的章节我们将看到如何更精确地定制列表视图以及行。

4.3.3 有序交互列表

本章前面曾提到可以使用 ol 代替 ul 来定义列表视图。我们已经知道，在只读列表上使用 ol 和 ul 的渲染效果是一样的，不过如果列表中使用的是交互行时就不一样了。

首先，一个重要的前提是这个列表中的所有行都必须是交互行，然后就可以使用 ol 了，这时各个链接会自动被编号，如图 4-11 所示。

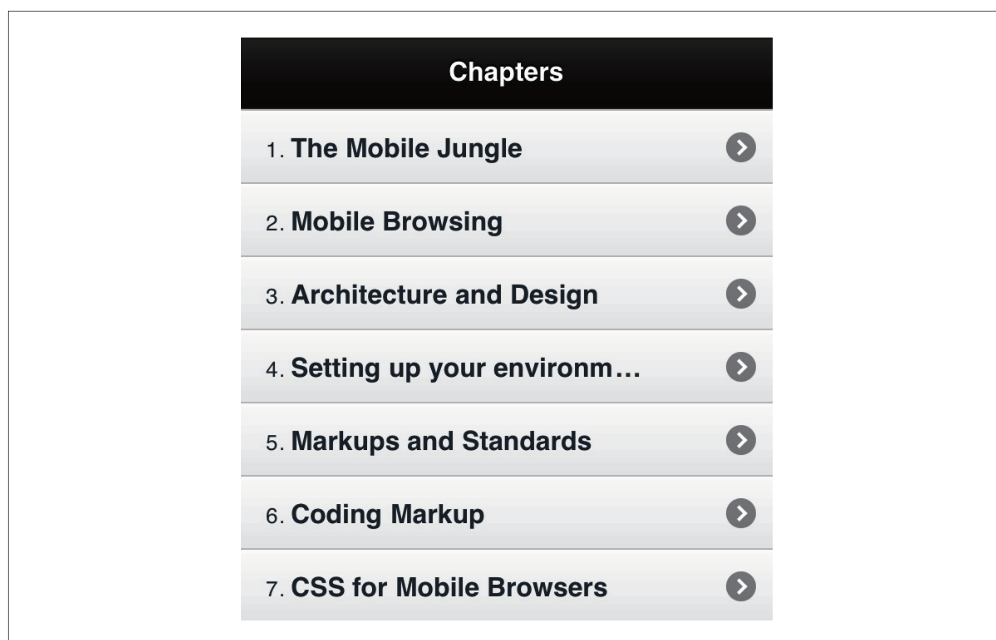


图 4-11：使用 ol 定义内容为交互行的列表视图，会得到如图所示的有序 UI

4.4 使用图片

每一行都可以用指定的图片来标识，并且可以添加两种不同的图片：图标和缩略图。

4.4.1　行图标

行图标是显示在行标题左侧的图片。不要把行图标与交互行右边的箭头或分割行的图标搞混了。

图标是一个位于对应 li 元素中的 16×16 像素的图片，由 ui-li-icon 类定义。例如：

```
<li>
   <img src="/icons/email.png" class="ul-li-icon">
   Send by e-mail
```

图标通常用于各种操作列表，例如对某条记录可以做的各种操作的列表（删除、编辑、通过电子邮件共享、通过社交网络共享等）。

4.4.2　缩略图

缩略图是一个 80×80 像素的图片，也位于文本左侧。在显示照片、图片或其他图形信息时，推荐每行都使用缩略图。

缩略图通常用于显示数据库记录列表，如朋友、图书、DVD、城市等。

缩略图由列表项中的图片定义，不需要任何特殊的类：

```
<li>
   <img src="/thumb/washington.png">
   George Washington
```

在图 4-12 中，可以看到两个插入列表中的图标及缩略图的例子。

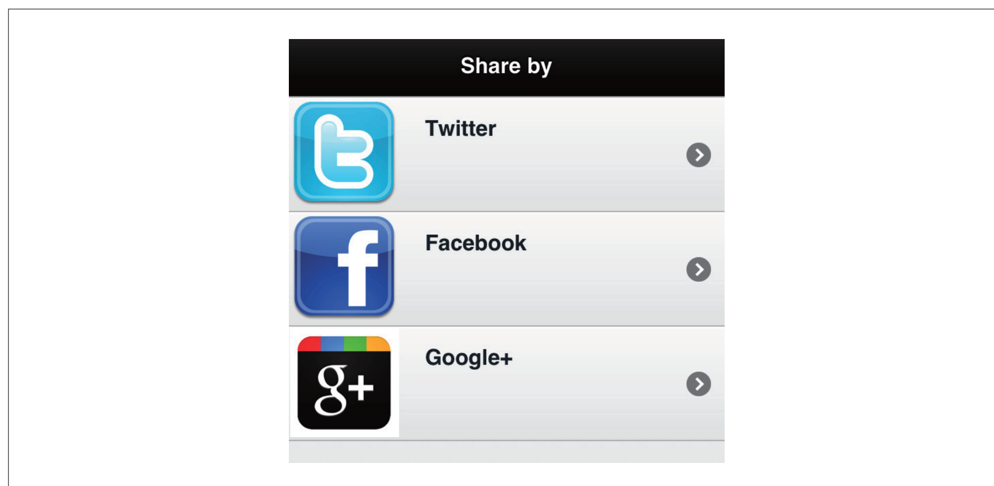


图 4-12：一图胜千言，所以我们为列表的每行都加了图标和缩略图

4.5 附加内容

到目前为止，我们设计的每个列表都只有一列文本内容。虽然可以添加缩略图或图标，但文本列只有一列。我们可以为每一行加一个次级列，用于显示补充信息。

这个功能可以使用任何带有 `ui-li-aside` 类的 HTML 元素来实现，比如 `span` 或 `div` 元素。

下面的例子在附加内容中显示了价格（如图 4-13 所示）：

```
<!DOCTYPE html>
<html>

<head>
  <!-- 这儿放置典型的 jQuery Mobile 头部 -->

</head>

<body>

   <div data-role="header">
     <h1>Order online</h1>
   </div>

   <div data-role="content">
      <ul data-role="listview">
          <li><a href="buy.html">Soda</a>
              <span class="ui-li-aside">$1.00</span>
          <li><a href="buy.html">Sandwich</a>
              <span class="ui-li-aside">$3.20</span>
          <li><a href="buy.html">Ice cream</a>
              <span class="ui-li-aside">$1.50</span>
      </ul>
   </div>

  </div>

</body>
</html>
```

图 4-13：使用附加内容可以在不使用新页面的情况下在一列中显示更多信息

4.6 标题与描述

如果想同时在一行上显示标题与描述，可以将标题放在 hx 标签里，将描述文本放在 p 元素里。它们不会被渲染为另外一行，效果如图 4-14 所示。

当然，也可以将标题、描述和附加内容混合使用，还能同时为那一行加上图标或缩略图。

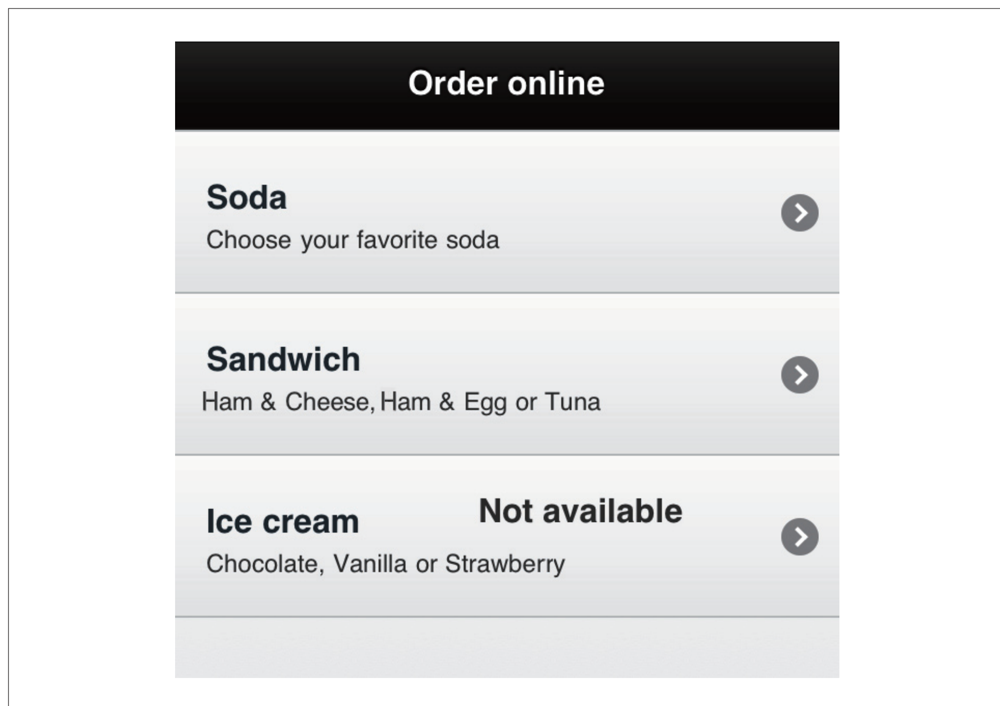
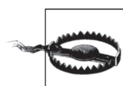


图 4-14：如果想在标题旁边显示一段较长的内容，最好的解决方案是使用标题和描述来显示，而不是使用附加内容

4.7 使用计数气泡

计数气泡是一个显示在行右侧的包含数字的圆圈，通常用于显示该交互行包含多少项，是一种非常简单的显示未读元素、未完成任务和任何其他数字信息的方式。



计数气泡并没有被限制为只能使用数字值，不过不建议使用文本，因为计数气泡的可用空间是专为数字值优化过的。对于文本值，可以使用附加内容或描述显示。

要使用计数气泡，只需在列表行中使用任意带 ui-li-count 属性的元素，如 span 标签，一切就完成了。例如：

```
<li><a href="inbox.html">Inbox</a>
    <span class="ui-li-count">86</span>
```

图 4-15 显示了计数气泡在一个交互列表中的渲染效果。如果想改变默认的白色背景，可以修改对应 ul 元素上的 data-count-theme 属性来指定计数气泡的色样。

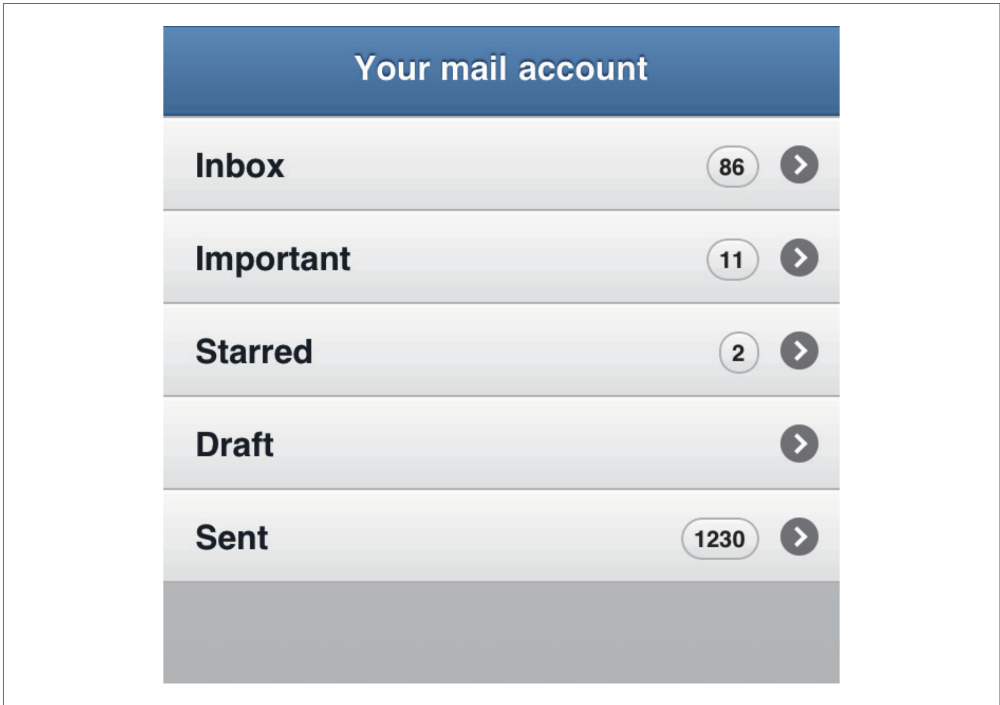


图 4-15：带计数气泡的交互列表

4.8 使用搜索过滤数据

最后来看看最精彩的部分。先把本书放下，如果你正在阅读本书的数字版，放下阅读器或离开浏览器，然后去看一看任何一个跟着本章完成的列表视图的示例。

找到对应的 ul 或 ol 元素，添加一个 data-filter="true" 属性，测试一下，然后回到这儿继续阅读。

jQuery Mobile 的魔法发生了。加了这个简单的属性之后，（整页或插入）列表顶部自动被加上了一个搜索框，并且这个搜索框真的能工作！

这个特性会根据用户的输入过滤列表元素。搜索文本框看起来非常漂亮（如图 4-16 所示），左侧带有一个搜索图标、一个水印提示文本，圆角，右侧还有一个清晰的按钮。

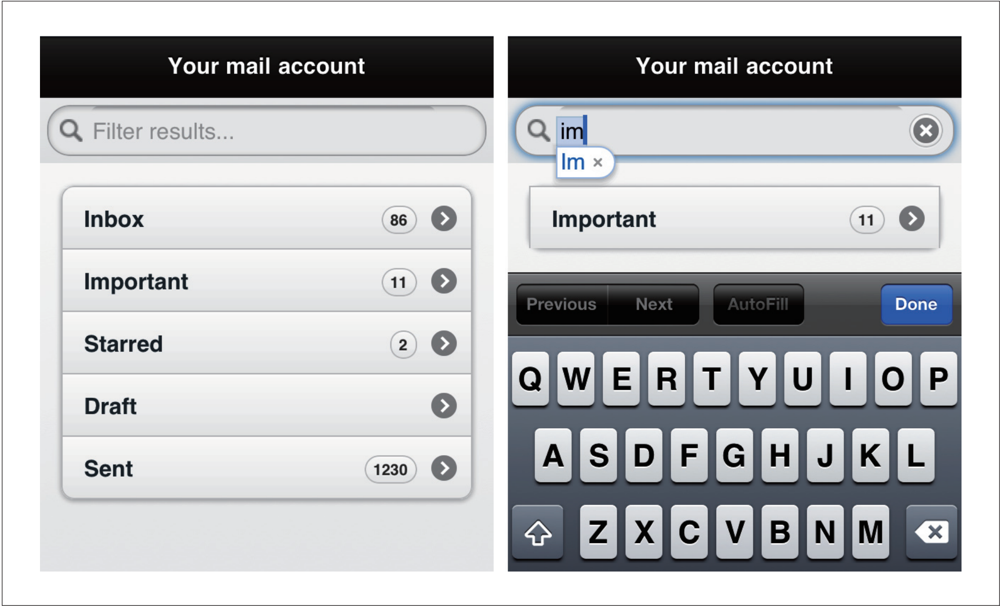


图 4-16：搜索过滤器是一个神奇的特性，无需编写代码即可为列表添加过滤系统

再说一次，要让这一切工作，只需使用下面的代码。现在请到本书的网站上获取对应的源码并复制粘贴——如果你都懒到了连自己输入以下代码都觉得麻烦的地步：

```
<ul data-role="listview" data-filter="true">
  <!-- 列表行 -->
</ul>
```

可以使用 `data-filter-placeholder` 属性来自定义占位文本，如：

```
<ul data-role="listview" data-filter="true"  data-filter-placeholder=
    "Search contacts...">
  <!-- 列表行 -->
</ul>
```

如果想自定义搜索栏的色样，则可以使用 `data-filter-theme` 属性。

4.9　列表视图速查表

前面几页中我们已经介绍了列表视图的许多特性。图 4-17 是一个速查表，可用于帮助理解如何使用 HTML 标签来渲染列表视图。

Item
Item
Item

Full-page list view

<ul data-role=”listview”>

Item
Item
Item

Inset list view

<ul data-role=”listview” data-inset=”true”>

Filter results... abc (X)
Item
Item

Filtered list view *(full-page or inset)*

<ul data-role=”listview” data-filter=”true”>

Item
Title >
1. Title >
Separator Title
Action #1 #2
Item
Item
Item
Title aside
Title count
Title
Description

Simple row Item

Interactive row Title *(list view must be ol)*

Numeric row Title

Separator row <li data-role=”list-divider”>

Splitted row Action #1 Action #2

Important row <hX>Item</hX>

Less important row <p>Item</p>

Thumbnail row <img src=”imageURL”>Item

Aside row *(can be interactive)* Itemaside

Count bubble row *(can be interactive)* Itemcount

Description row <hX>Title</hX> <p>Description</p>

图 4-17：列表视图特性一览

第 5 章

表单组件

jQuery Mobile 框架支持标准网页表单，在支持的设备上会自动使用 AJAX 处理，同时标准表单控件的外观也为触摸操作做了优化。这是关于表单控件的第一个好消息。

上面所说的标准网页表单，指的是位于 `form` 元素内的一组表单控件，如 `input`、`textarea` 以及 `select` 元素等，表单的数据将被发送到该 `form` 元素的 `action` 属性所指向的 URL。

5.1 表单动作

所有表单 `action` 的处理方式都与原来一样。也就是说，当用户按下提交按钮或回车键时，表单会被提交。我刚刚说了移动设备上的回车键了吗？

我们的目标是兼容各种设备以及各种输入方式，其中一些有物理键盘（可能带回车键），一些只有一个虚拟的屏幕键盘。那些设备的虚拟键盘上通常会有一个“发送”、“提交”之类的按钮，用于提交表单。

表单提交时，除非是提交到不同的域名，不然 jQuery Mobile 会使用 AJAX 方式过渡到目标页面，就像连接到一个外部页面一样。表单 `form` 元素的 `method` 属性可以是 `get`，也可以是 `post`。

典型的 jQuery Mobile 表单与典型的网页表单很像：

```
<form action="send.php" action="get">

</form>
```

jQuery Mobile 表单要求结果页面也是一个 jQuery Mobile 文档，就像外部页面一样。这是因为默认情况下，jQuery Mobile 会尝试通过框架中的 AJAX 发送表单。

如果想将结果页面添加到浏览器历史记录中，可以使用 get 方法并修改页面地址的散列值。

第二章中讲到的 data-transition 和 data-direction 属性也可以应用到 form 元素上。因此，可以定义一个弹出式过渡的表单提交行为：

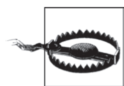
```
<form action="send.php" action="get" data-transition="pop">

</form>
```

强制表单禁用AJAX

如果想强制使用标准 HTTP 请求而不是 AJAX，可以在 form 元素上使用 data-ajax="false" 属性。在表单 action 指向不同域的主机或在上传文件时（后面会详细介绍），这个属性设置尤其有用。

也可以在 form 元素上使用 target="_blank" 来强制使用非 AJAX 方式提交。



可以在表单域上定义一个布尔值的属性 autofocus，以便控制该表单域在页面加载后是否自动获得焦点。不过在 jQuery Mobile 文档中，这个属性只会在第一个页面上生效，之后使用 jQuery Mobile 过渡新加载的页面都没有效果。

5.2 表单元素

jQuery Mobile 允许在一个表单中同时使用标准网页表单控件和新的富控件。jQuery Mobile 框架有一个称为“自动初始化”（auto-initialization）的特性，可以将各个网页表单控件替换为对触摸操作更为友好的富控件。

jQuery Mobile 也让 HTML5 中新增的表单域类型达到了一个新的层次，甚至在不支持这些新表单域的浏览器上也可以使用它们。

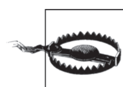
下列元素会被渲染为富控件：

- 按钮，使用 button 或 input 元素；

- 文本输入域，使用 `input` 或 `textarea` 元素；
- 复选和单选按钮，使用 `input` 元素；
- 菜单，使用 `select` 及 `option` 元素；
- 滑块，使用 `input type="range"` 的新控件；
- 滑块开关，使用带新 `role` 属性的 `select` 和 `option` 元素。

如果想强制禁止 jQuery Mobile 使用富 UI 组件来渲染表单控件，只需在各个表单元素上加上 `data-role="none"` 属性。

每个表单元素都会独占一行，对移动设备上的表单来说，这是用户体验最好的方式。当然，也可以强制使用多列，不过不建议这么设计用于移动设备的表单。



使用 AJAX 时，包括网页表单控件在内的整个页面都共享同一个 DOM，因此我们需要保证各个 `form` 元素的 ID 在整个站点中唯一。如果有两个表单，也不要在相似的表单控件上使用相同的 ID。

5.2.1　文本标签

每个表单控件都有一个很重要的文本标签元素。应该总是为每个控件包含一个对应的 `label` 元素，并将控件的 `for` 属性指向该 `label`。来看一个例子：

```
<label for="company">Company Name:</label>
<input type="text" id="company">
```

关于 `label` 需要了解的最重要的一点是用户点击 `label` 时，对应的表单控件会得到焦点，用户可以马上使用该控件。因此，表单元素可点击的区域为控件自身加上 `label` 标签，对触摸设备来说这是不错的体验。



如果出于某种原因，需要创建一个不带 `label` 的表单元素时，你仍然要明确地考虑一下可访问性问题。这也是为什么总是应该包含一个 `label`，如果想隐藏它，可以在它或任何其他 HTML 元素上应用类 `ui-hidden-accessible`。

5.2.2　域容器

域容器是一个可选的文本标签 / 部件包装器，用于提高在平板电脑等宽屏幕设备上的体验。容器可以是任意带 `data-role="fieldcontainer"` 的块级元素。

容器会将内部的文本标签和表单控件自动对齐，并会添加一个细边框作为字段分隔

符。容器的第二个重要特性是它会根据设备的宽度自动调整布局。也就是说，如果设备宽度比较窄，文本标签就会放置在表单控件顶部，否则的话就使用两列布局，文本标签位于左列，如图 5-1 所示。

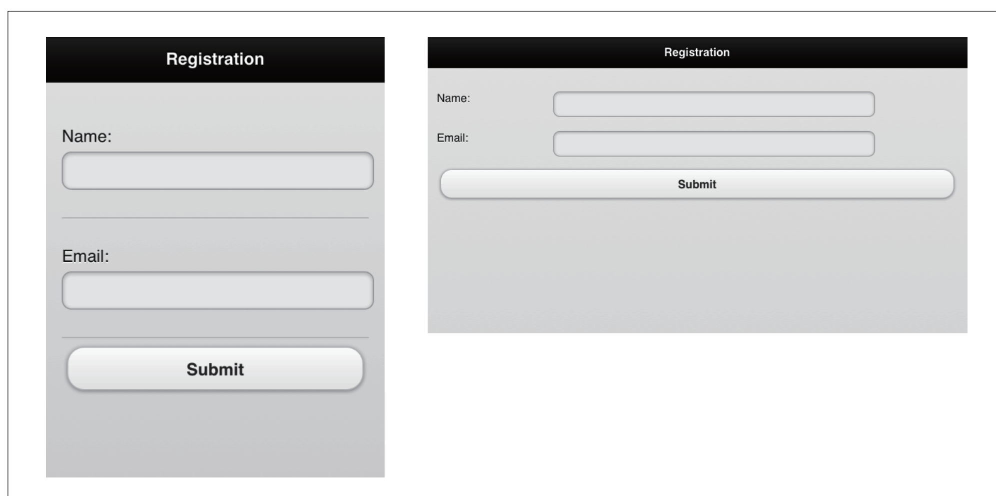


图 5-1：基于广为人知的移动设计模式，文本标签的位置在智能手机和平板电脑上有所不同

每个文本标签 / 控件都可以添加域容器，如下例所示：

```
<div data-role="fieldcontainer">
<label for="company">Company Name:</label>
<input type="text" id="company" name="company">
</div>
<div data-role="fieldcontainer">
<label for="email">Email:</label>
<input type="email" id="email" name="email">
</div>
```

5.2.3　文本输入框

jQuery Mobile 支持基本的 HTML5 文本输入控件，并能根据当前主题和色样进行渲染。下面是可用的文本输入框。

- `<input type="text">`
- `<input type="password">`
- `<input type="email">`
- `<input type="tel">`
- `<input type="url">`
- `<input type="search">`

- <input type="number">
- <textarea>

使用上面的元素时会自动得到一个 jQuery Mobile 控件。记住不需要显式指定任何 data-role 属性。



type 为 button、submit、clear 或 image 的 input 元素会被自动渲染为 jQuery Mobile 按钮。

如果你正在想“我只知道 text 和 password 两种输入类型”的话，不要担心。email、tel、url、search 以及 number 是 HTML5 标准中的新输入类型，在移动互联网领域非常重要。你知道，大多数触摸设备没有物理键盘，指定一种新输入类型时，我们将得到一个优化过的虚拟键盘，如图 5-2 所示。

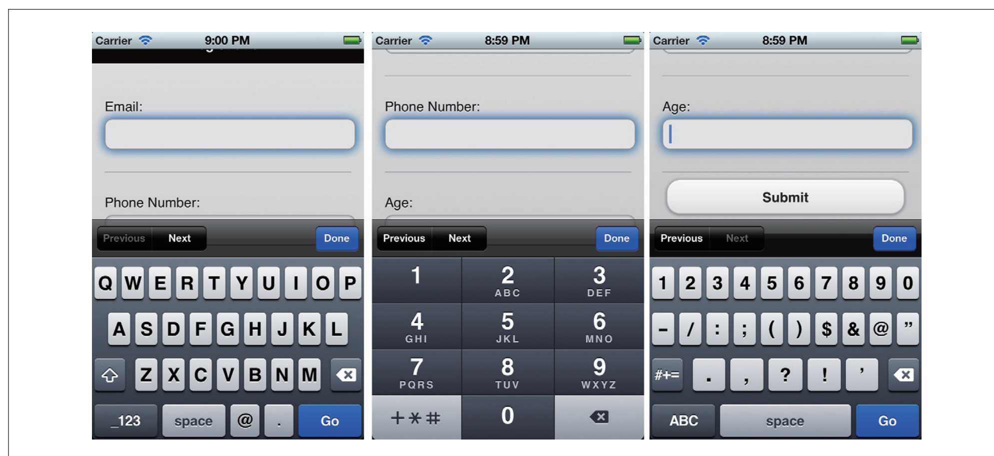


图 5-2：在一些设备上，使用新的 HTML5 输入类型会得到不同的虚拟键盘



不要担心那些不支持 HTML5 新输入类型的老设备，遇到不认识的 type 属性时，每个浏览器都会将它降级为基本的文本输入域。

与典型的 text 类型相比，search 类型的输入域有两个不同：在 jQuery Mobile 中它的用户界面与其他输入域不同，在某些设备的虚拟键盘上它提供了一个不同的“返回”键。如图 5-3 所示：

```
<div data-role="fieldcontainer">
<label for="search">Search:</label>
<input type="search" id="search" name="search">
</div>
```

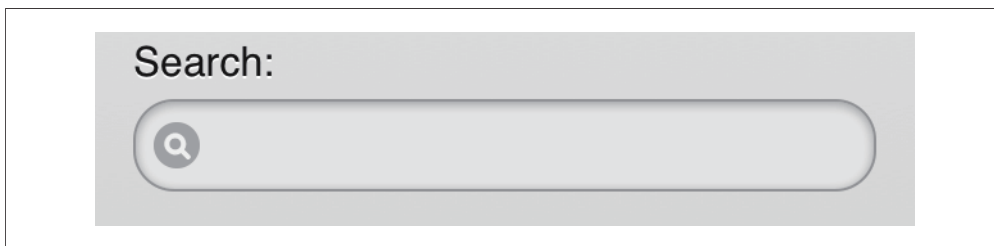


图 5-3：在 jQuery Mobile 文档中，`search` 输入域的外观有所不同

5.2.4　自增长文本区

使用 `textarea` 来处理多行文本输入时，有一个额外的特性：自动增长。jQuery Mobile 会以一个两行高的区域开始，随着输入文本的增加而变高。之后，框架会自动扩展该文本区域以适合新行。这将允许我们在 `textarea` 中滚动，而这个操作在大多数移动浏览器上都是非常痛苦的。

```
<div data-role="fieldcontainer">
<label for="comments">Your comments:</label>
<textarea id="comments" name="comments></textarea>
</div>
```

图 5-4 显示了自增长文本区域的实例。

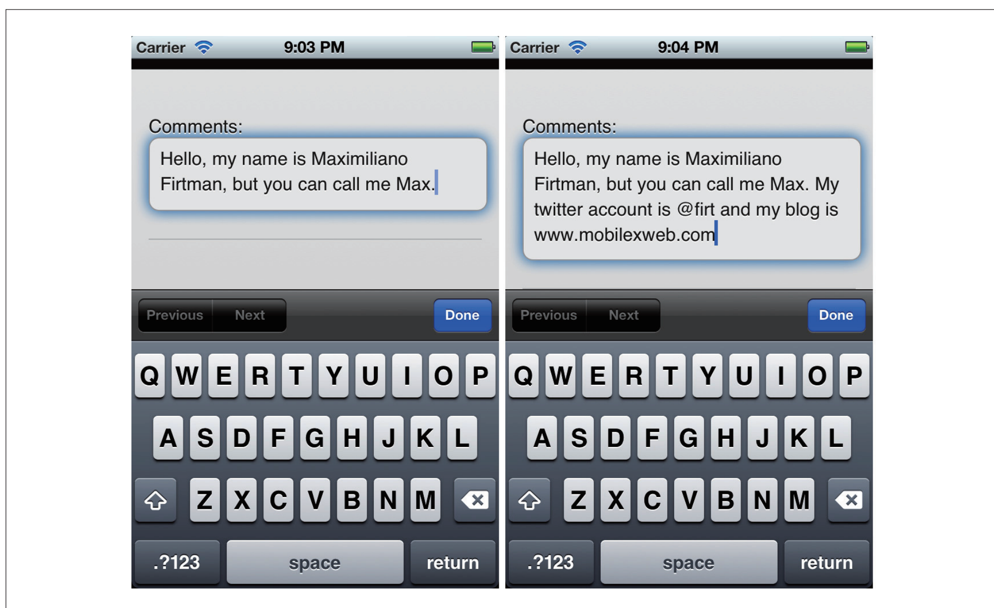


图 5-4：当文本行数超出可见区域时，jQuery Mobile 会自动扩展文本区域

安全还是不安全

在移动设备上使用密码文本框（`<input type="password" />`）是一个有争议的话题。密码文本框（使用经典的星号或圆点代替用户输入的字符）的初衷是避免站在用户背后或能看到用户屏幕的人窃取用户的密码。在移动电话领域，这个情况发生了变化。由于屏幕及字体的尺寸都很小，因此其他人很难看见用户在自己的移动电话上输入的内容。进一步，在非QWERTY键盘的设备上打字是比较困难的，如果用星号来代替真实的输入字符，用户可能无法确定自己的输入是否正确（虽然在有些设备上，输入的字符会先显示一秒再变成星号）。如果仍然想使用密码输入框，建议强制将文本输入框改变为数字输入框。互联网可用性专家Jakob Nielsen（http://useit.com）就同意这一点。在2009年Alertbox专栏上，Nielsen写道："用户输入密码时，唯一的反馈是看到一行圆点，可用性实在是太糟糕了。一般情况下，遮住密码不会提高安全性，只会由于登录失败而增加成本。"Facebook移动版部署了这样一个特性，如果输错了一次密码，下次输入时密码输入框的内容就将是可见的，同时还有一个说明告诉用户密码虽然可见，但它仍然是安全的。另一个解决方案是使用普通文本框来输入密码，同时再加一个文本标签为"隐藏密码"的复选框，选中时就将密码输入框的类型更改为`password`。

5.2.5 新HTML5属性

在文本输入框上使用任何 HTML5 属性都是安全的，在支持的设备上这些属性将起作用，老设备上则将什么也不做。在新的表单控件属性里，我们会提到 `required`、`pattern`、`placeholder` 以及 `min` 和 `max`（仅用于 `<input type="number">`）。

因此，如果一个输入域是必需的，则可以使用布尔属性 `required` 来标识它。还可以定义一段占位文本，当输入控件中没有值时，它将显示为灰色的提示。

```
<div data-role="fieldcontainer">
<label for="name">Your Name:</label>
<input type="text" id="name" required placeholder="Enter your name">
</div>

<div data-role="fieldcontainer">
<label for="age">Your Age:</label>
<input type="number" id="age" required placeholder="Enter your age" min=
    "10" max="110">
</div>
```



使用 CSS3 的伪类，无需编写 JavaScript 校验代码即可为有效、无效、必填以及选填的文本输入框创建不同的样式。

5.2.6 日期输入框

在 HTML 中输入日期一直是一个问题，过去一般需要依赖于 JavaScript 库来用 HTML 创建一个虚拟的日历。HTML5 中已加入了对日期输入框的支持，只需要使用下列类型的 `input` 元素：

- `date` 用于日期选择（年、月、日）；
- `datetime` 用于完整的日期选择（年、月、日、小时、分钟），使用标准的包括 GMT 时区的语法；
- `time` 用于时间选择（小时、分钟）；
- `datetime-local` 用于完整的日期选择，不带时区信息；
- `month` 用于月份选择（一般渲染为一个下拉列表）；
- `week` 用于选择指定年份的星期（一般渲染为一个下拉列表）。

日期输入类型是新添加的，不是所有的移动或桌面浏览器都支持。在写作本书的时候，大多数日期输入类型在 iOS 5.0 版（以及之后的版本）中的 Safari 上都能工作，同时支持的还有用于 PlayBook 和智能手机的 BlackBerry 浏览器 5.0 及更新版、Opera 移动版，以及 Firefox Android 版。可以在 http://mobilehtml5.org 上查看这个兼容的更新情况。



即使没有显式地指定 `data-role`，还是要记住输入框控件是被渲染为 jQuery Mobile 控件的。因此，用于指定色样的 `data-theme` 等全局属性对这些控件也有效果。

记住，在不支持的浏览器上，`input` 元素会自动降级为普通的文本输入框：

```
<div data-role="fieldcontainer">
<label for="birth">Your Birthdate:</label>
<input type="date" id="birth" name="birth">

<label for="favmonth">Your favorite month:</label>
<input type="month" id="favmonth" name="favmonth">
</div>
```

日期输入框支持 `min` 和 `max` 属性。在图 5-5 中可以看到日期输入框在一些智能手机和平板电脑上的渲染效果。

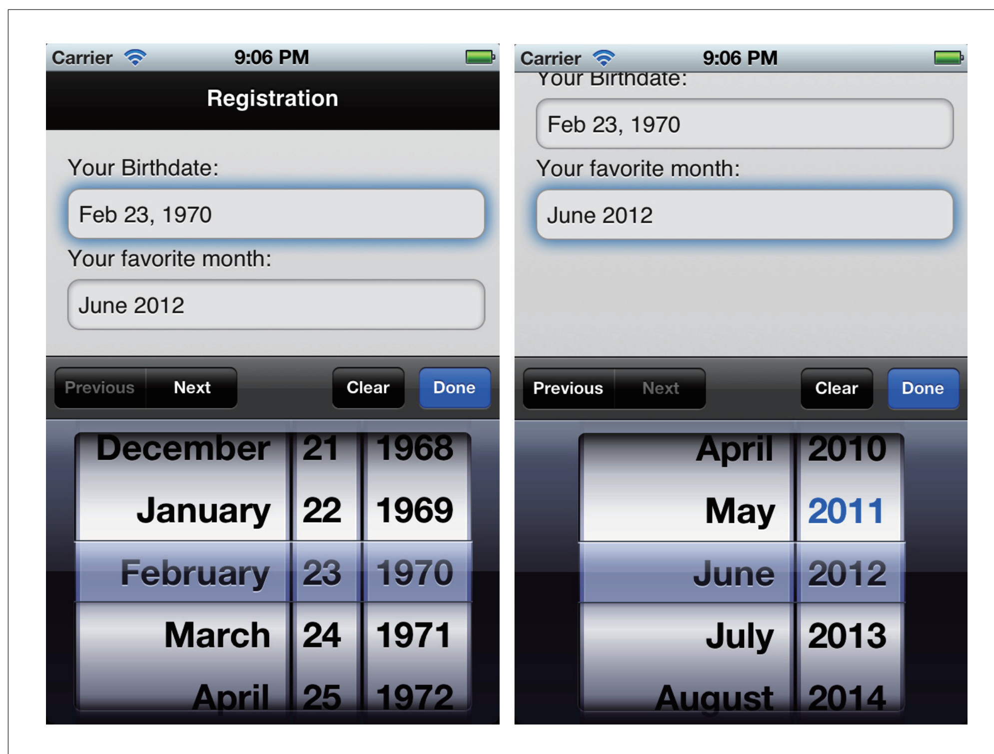


图 5-5：iOS 从第 5 版开始支持日期元素，因此，使用日期 HTML 输入框类型可以得到更好的日期选择器

5.2.7 滑块

滑块用于输入处于某个范围的数字值。使用时，它会提供一个接受数字的文本输入框，右侧还会有一个水平滑块，如图 5-6 所示。要使用滑块，必须定义一个 HTML5 `<input type="range">` 控件，它接受 `min`、`max` 以及 `step` 值：

```
<div data-role="fieldcontainer">
<label for="qty">Quantity:</label>
<input type="range" id="qty" name="qty" min="1" max="100" value="5">
</div>
```

滑块支持通过 `data-theme` 定义色样，但只对数字输入框有效，也可以定义一个 `data-track-theme` 属性，后者只影响滑动条，如图 5-6 所示：

```
<div data-role="fieldcontainer">
<label for="qty">Quantity:</label>
<input type="range" id="qty" name="qty" min="1" max="100" value="5" data-
    theme="e" data-track-theme="b">
</div>
```

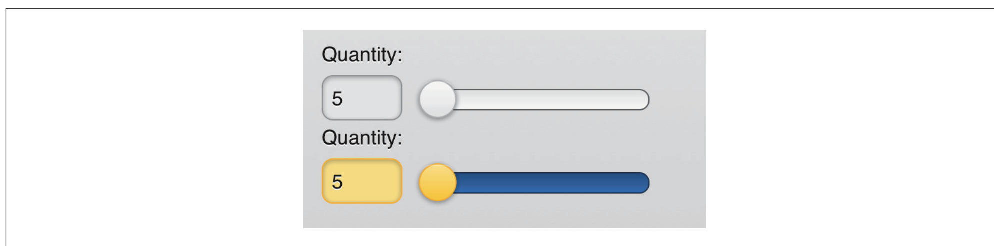


图 5-6：在 jQuery Mobile 中，范围输入类型将在标准 HTML5 渲染的基础上被渲染为一个只能输入数字的小输入框和一个同步的水平滑块

如果设备支持，滑块控件也能响应键盘事件。因此，当控件获得焦点时，用户可以使用方向键、滚轮，或操纵杆来增加或减少控件的值。

5.2.8 平移切换开关

平移切换开关是一个布尔值（true 或 false，on 或 off）选择器，功能上与复选框类似，但用户界面完全不同。它被渲染为一个可视化的开关，用户可以打开或关闭（在开关上点击或拖动）。

这是第一个需要显式指定 data-role 的表单控件，对应的 data-role 值为 slider。它需要一个 select 元素，其中只包含两个 option 作为子元素，第一个为 off/false 值，第二个是 on/true 值：

```
  <label for="updated">Receive updates</label>
<select id="updated" name="updated" data-role="slider">
<option value="no">No</option>
<option value="yes">Yes</option>
</select>
```

不指定域容器的话，平移切换开关将被渲染为整页宽度。更常见的情况是将相应的内容放在域容器里，如图 5-7 所示。

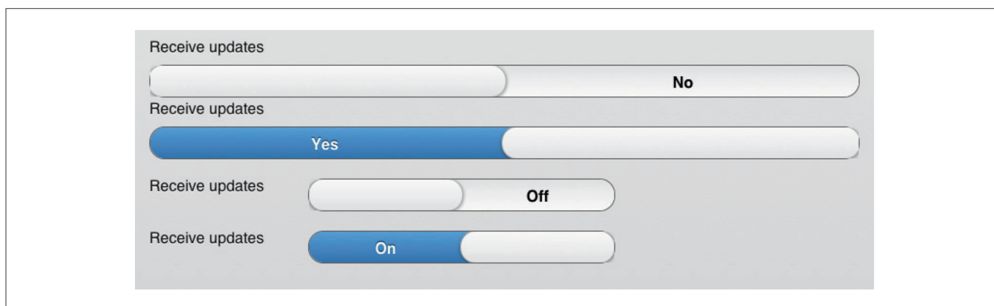
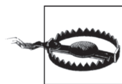


图 5-7：使用域容器时，平移切换开关的宽度会不一样



HTML5 拾色器控件 `<input type="color">` 在大多数移动浏览器上都不能工作，渲染效果和任何其他文本输入控件一样。

5.2.9 选择菜单

select 元素创建的菜单是一个典型的表单控件，用于从一个弹出列表中选择一个或多个选项。所有移动浏览器都支持 select（单选或多选）。jQuery Mobile 将选择菜单的外观改变为按钮样式的风格，并在被点击时调用原生的菜单，如图 5-8 所示。

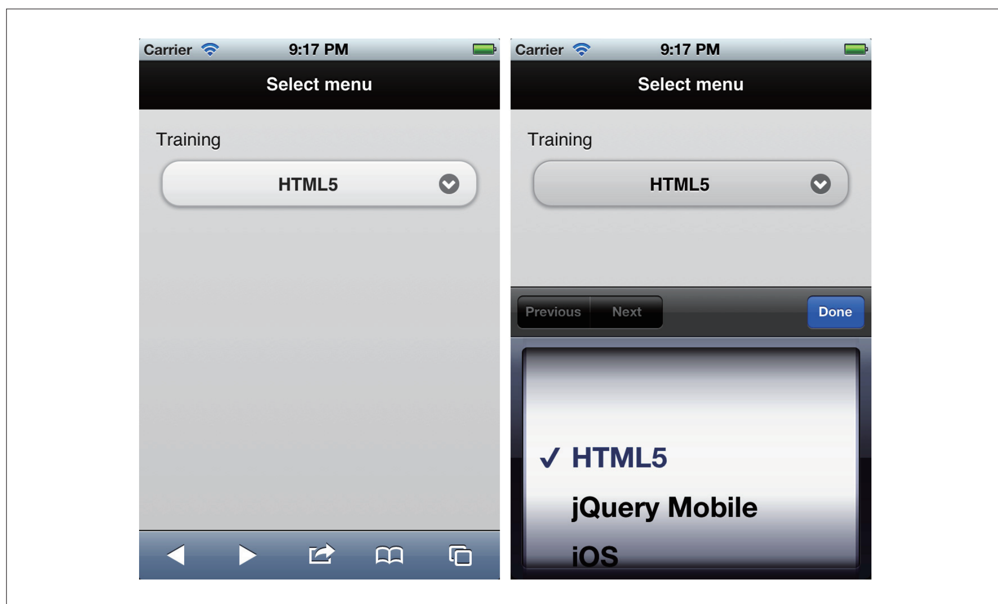


图 5-8：select 菜单会被自动转换为像按钮一样的控件，并带有一个提示用户可以打开选项菜单的图标

默认情况下，select 菜单会占用整个宽度，除非将它包含在一个域容器里：

```
  <label for="training">Training</label>
<select id="training" name="training">
<option value="1">HTML5</option>
<option value="2">jQuery Mobile</option>
<option value="3">iOS</option>
<option value="4">Android</option>
<option value="5">BlackBerry</option>
<option value="6">Qt for Meego</option>
</select>
```

使用 multiple 布尔属性后，jQuery Mobile 及原生控件都将提供另一种用于多选的用户界面，如图 5-9 所示：

```
  <label for="lang">Languages you like</label>
<select id="lang" name="lang" multiple>
<option value="1">C/C++</option>
<option value="2">Objective-C</option>
<option value="3">Java</option>
<option value="4">C#</option>
<option value="5">Visual Basic</option>
<option value="6">ActionScript</option>
<option value="7">Delphi</option>
<option value="8">Phyton</option>
<option value="9">JavaScript</option>
<option value="10">Ruby</option>
<option value="11">PHP</option>
</select>
```

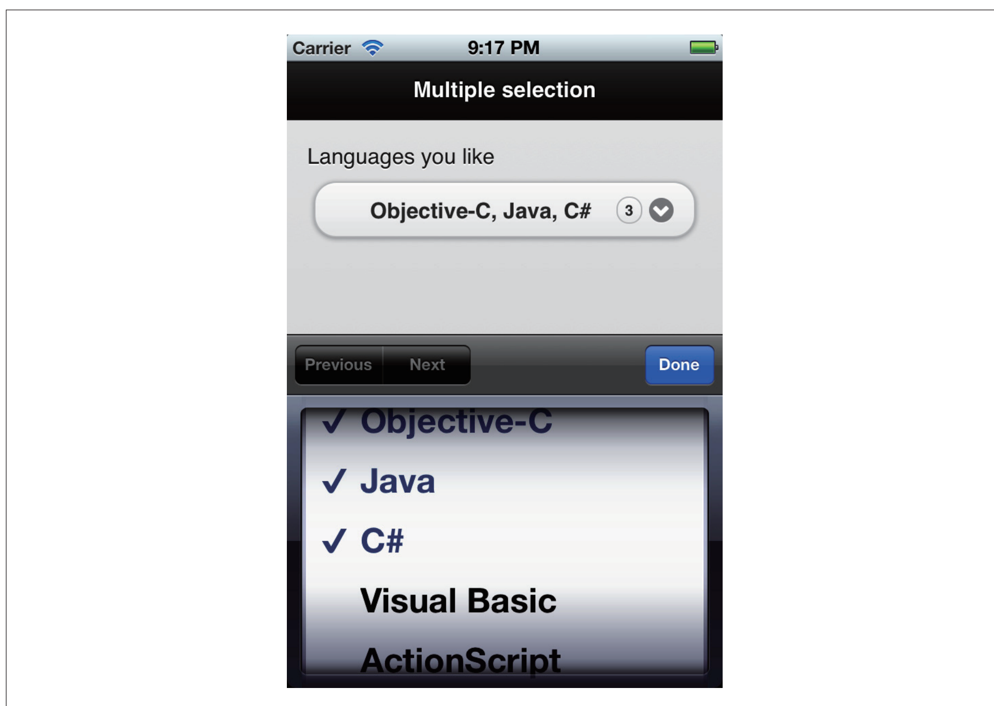
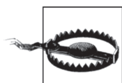


图 5-9：使用多选菜单时，界面会显示使用逗号分隔的已选中元素以及一个显示已选中多少元素的计数气泡

和其他表单元素一样，也可以使用 data-theme 来改变默认的色样，还可以用 optgroup 元素将 option 元素分组。



如果使用 JavaScript 改变表单控件的值，例如 select 菜单的 value 属性，或单选、复选框的 checked 属性，对应的 jQuery Mobile UI 不会自动更新，除非使用下一章介绍的 jQuery Mobile API 刷新对应的部件。

1. 组合选择菜单

select 菜单可以使用 controlgroup 元素进行垂直或水平组合。水平组合对较短的选择菜单很有用，例如一组年 / 月 / 日的选择。要组合菜单，只需将它们嵌入一个带 data-role="controlgroup" 的 div 中，如下例所示，效果见图 5-10：

```
<div data-role="controlgroup">
<legend>Color and Size</legend>
<select id="color" name="color">
<option value="1">Blue</option>
<option value="2">White</option>
<option value="3">Red</option>
<option value="4">Black</option>
<option value="5">Pink</option>
</select>
<select id="size" name="size">
<option value="1">X-Small</option>
<option value="2">Small</option>
<option value="3">Medium</option>
<option value="4">Large</option>
<option value="5">X-Large</option>
</select>
</div>

<div data-role="controlgroup" data-type="horizontal">
<legend>Week day and time</legend>
<select id="weekday" name="weekday" multiple>
<option value="1">Mon</option>
<option value="2">Tue</option>
<option value="3">Wed</option>
<option value="4">Thu</option>
<option value="5">Fri</option>
</select>
<select id="time" name="time">
<option value="1">Morning</option>
<option value="2">Midday</option>
<option value="3">Afternoon</option>
</select>
</div>
```

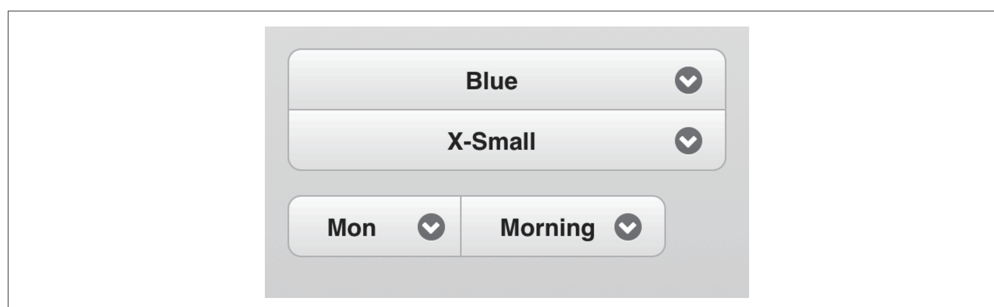


图 5-10：可以使用 **controlgroup** 包装将选择菜单组合起来

使用 controlgroup 元素时，单独的文本标签（label）会被隐藏。可以使用 legend 元素来定义一个应用到整个分组的文本标签，如下例所示，效果见图 5-11：

```
<div data-role="controlgroup" data-type="horizontal">
<legend>Delivery options</legend>

<label for="weekday">Week day</label>
<select id="weekday" name="weekday" multiple>
<option value="1">Mon</option>
<option value="2">Tue</option>
<option value="3">Wed</option>
<option value="4">Thu</option>
<option value="5">Fri</option>
</select>
<label for="time">Time</label>
<select id="time" name="time">
<option value="1">Morning</option>
<option value="2">Midday</option>
<option value="3">Afternoon</option>
</select>
</div>
```

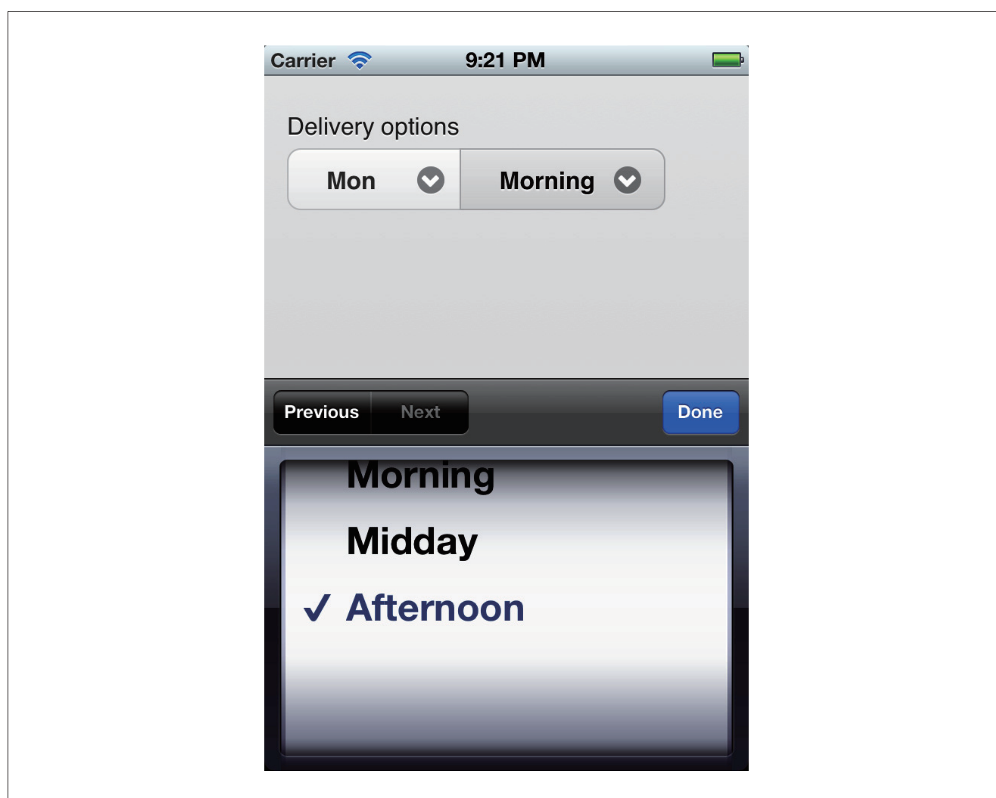


图 5-11：对不是太长的选择菜单来说，可以使用水平的 controlgroup 元素

2. 非原生选择菜单

在图 5-10 中我们已经看到，每个 `select` 元素都默认被渲染为一个 jQuery Mobile 富控件。不过在被打开时，它仍然依赖于移动浏览器的原生 `select` 元素。jQuery Mobile 也为选择菜单提供了另一个用户界面，可用于覆盖原生行为。

要激活这个特性，只需在对应的 `select` 元素上指定 `data-native-menu="false"`。如图 5-12，在菜单关闭时 UI 看起来没什么不同，不过打开时就能看到差异了：

```
  <label for="training">Training</label>
<select id="training" name="training" data-native-menu="false">
<option value="1">HTML5</option>
<option value="2">jQuery Mobile</option>
<option value="3">iOS</option>
<option value="4">Android</option>
<option value="5">BlackBerry</option>
<option value="6">Qt for Meego</option>
</select>
```

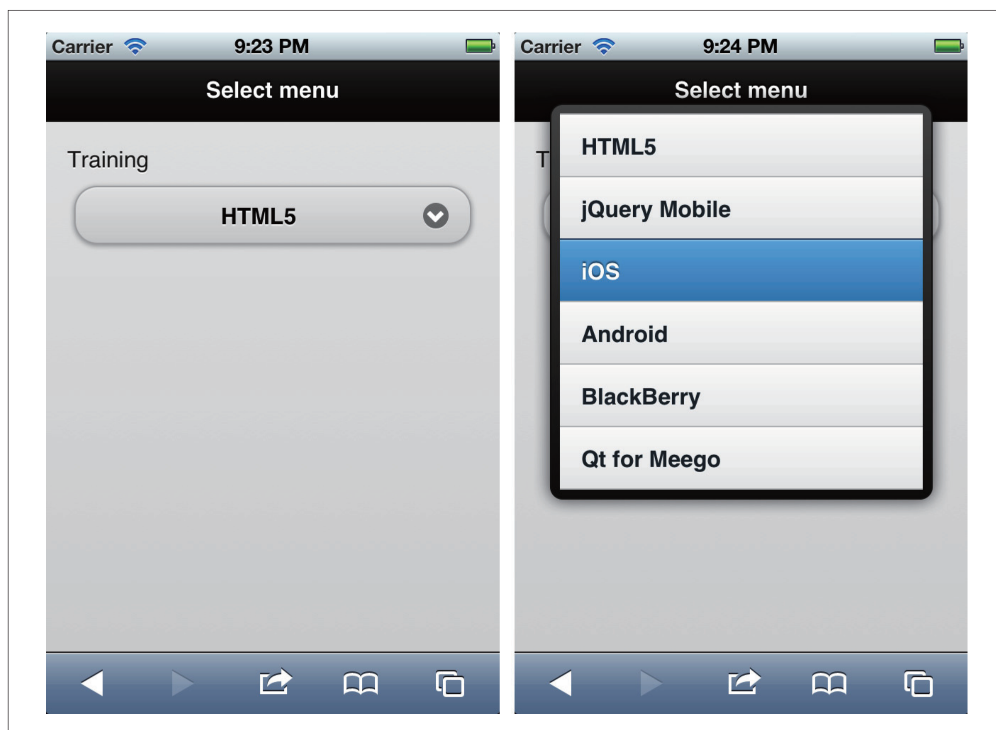


图 5-12：使用非原生选择菜单时，原来的菜单会被一个类似对话框的交互列表取代

当列表的长度超过屏幕时，选择菜单控件的外观会发生变化，它将使用一个类似对话框的外观，如图 5-13 所示。

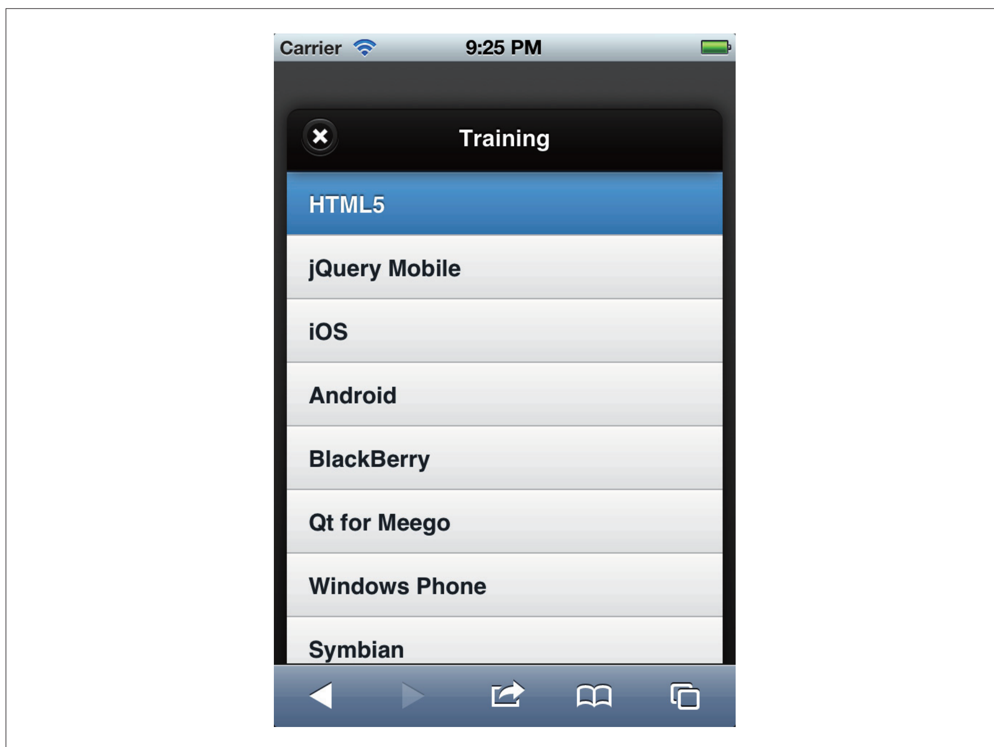


图 5-13：如果选择菜单的选项太多，打开菜单时将创建一个对话页面



使用非原生菜单时，optgroup 元素会被渲染为列表视图中的列表分割行。

打开非原生菜单时当前表单会被盖上一个覆盖层，可以用 data-overlay-theme 来指定这个覆盖层的色样。如果某个 option 元素显式地指定了空值（value=""），或带有属性 data-placeholder="true"，则它将被用作覆盖层的标题，而不再是一个可选择的选项，如下例所示（参见图 5-14）：

```
  <label for="training">Training</label>
<select id="training" name="training" data-native-menu="false" data-
overlay-theme="e">
<option value="0" data-placeholder="true">Select your training</option>
<option value="1">HTML5</option>
<option value="2">jQuery Mobile</option>
<option value="3">iOS</option>
<option value="4">Android</option>
<option value="5">BlackBerry</option>
<option value="6">Qt for Meego</option>
</select>
```

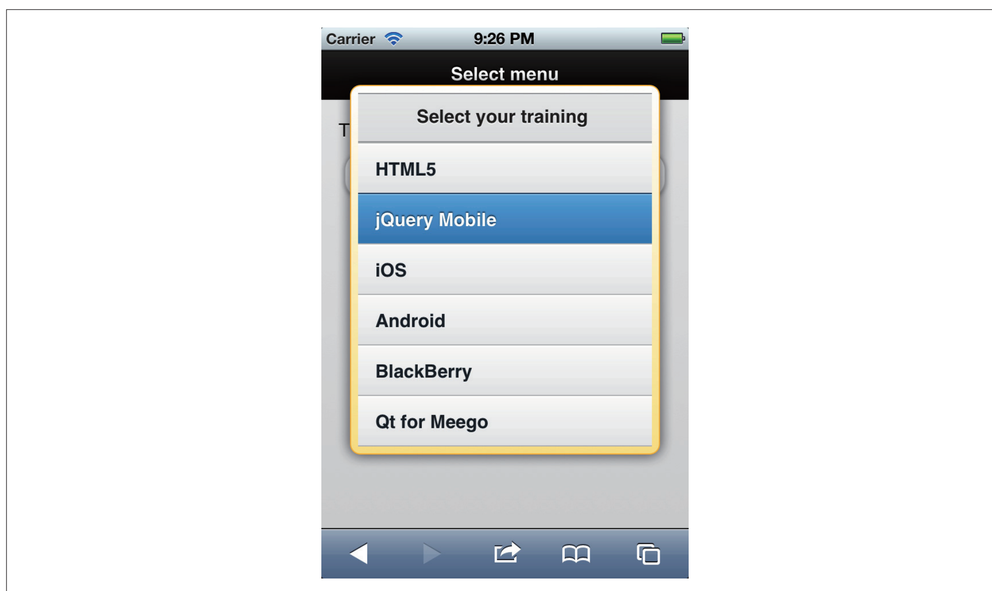


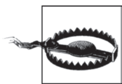
图 5-14：在非原生选择列表中，可以通过占位符来指定选择对话的标题

如图 5-15 所示，非原生菜单也支持多选，只需指定布尔值属性 `multiple` 即可。指定了 `multiple` 时，该覆盖层对话将带一个关闭按钮，并且不会在选择了一项之后就自动关闭。



图 5-15：非原生多选菜单的用户界面包含一个多选列表视图

jQuery Mobile 也会正确地处理 option 元素的 disabled 属性。



要让非原生菜单正常工作，在 option 元素上有一个限制：它们都必须显式地指定 value 属性。如果一个 option 的 value 被指定为空，那么它将被用作弹出窗口的标题。

5.2.10 单选按钮

大家都知道单选按钮是什么。在 jQuery Mobile 中，关于单选按钮最好的一个消息是，对它们的渲染我们什么也不需要做。首先，让我们看一下要让单选按钮在 jQuery Mobile 中正常工作需要些什么条件：

- 每个选项必须是一个 <input type="radio">；
- 同一组的每个选项的 name 值必须相同；
- 每个选项都必须有一个唯一的 id 以及一个关联的唯一的 label 元素。

使用选择菜单和单选按钮的一个很大的不同是，使用 select 时对应的 label 是应用到整个元素上，而使用单选按钮时对应的 label 则是应用到每个元素上。在图 5-16 中我们可以看到，label 是按钮文本的一部分，而不像文本输入框控件的 label 一样位于控件之外。

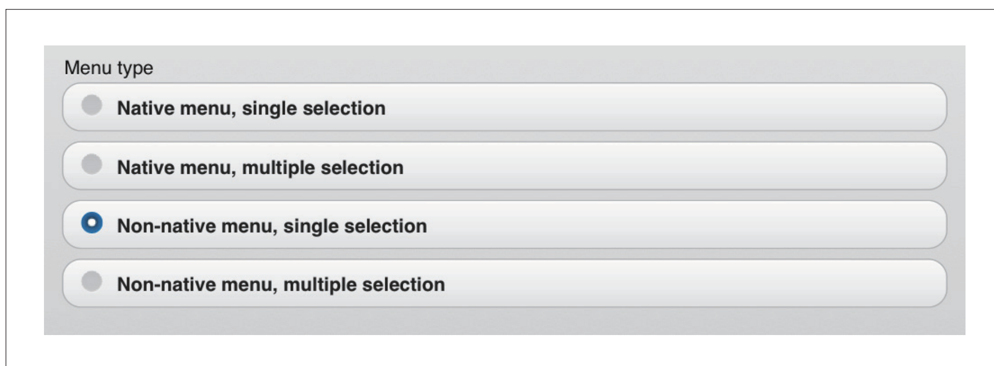


图 5-16：单选按钮与对应的 label 被渲染在同一个按钮里

也可以使用 HTML 元素 legend 来为整个选项组提供一个文本标签。

来看一个例子：

```
<legend>Menu type</legend>

<label for="menuNative1">Native menu, single selection</label>
```

```
<input type="radio" id="menuNative1" name="menuType" value="1">

<label for="menuNative2">Native menu, multiple selection</label>
<input type="radio" id="menuNative2" name="menuType" value="2">

<label for="menuNonNative1">Non-native menu, single selection</label>
<input type="radio" id="menuNonNative1" name="menuType" value="3">

<label for="menuNonNative2">Non-native menu, multiple selection</label>
<input type="radio" id="menuNonNative2" name="menuType" value="4">
```

如果将每个 radio 元素都包含在 controlgroup 容器中，用户界面会更友好一些，如图 5-17 所示：

```
<div data-role="controlgroup">
<!-- 这儿放置 label 和 radio 元素 -->
</div>
```

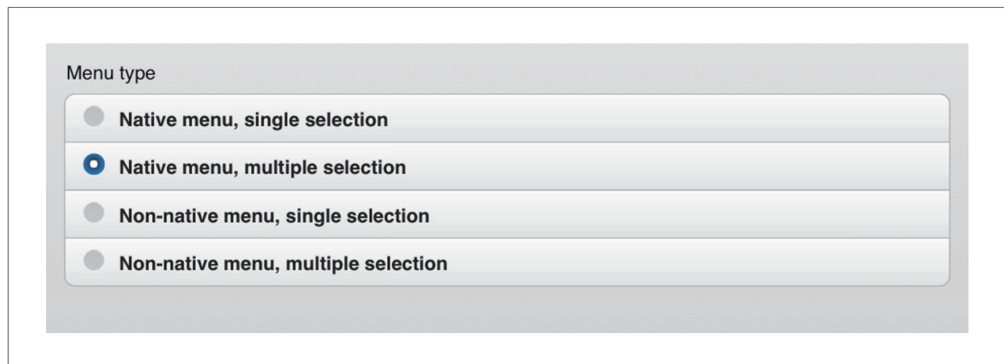


图 5-17：将多个单选按钮组合到一个 **controlgroup** 中是提供清晰的用户界面的最佳方法

如果将 controlgroup 元素的 type 改为 horizontal，则外观将大不一样。元素将不再是典型的单选按钮，而会变成开关按钮控件，如图 5-18 所示：

```
<legend>Delivery method</legend>

<div data-role="controlgroup" data-type="horizontal">

<label for="deliveryUPS">UPS</label>
<input type="radio" id="deliveryUPS" name="delivery" value="ups">

<label for="deliveryDHL">DHL</label>
<input type="radio" id="deliveryDHL" name="delivery" value="dhl">

<label for="deliveryFedex">FedEx</label>
<input type="radio" id="deliveryFedex" name="delivery" value="fedex">

</div>
```

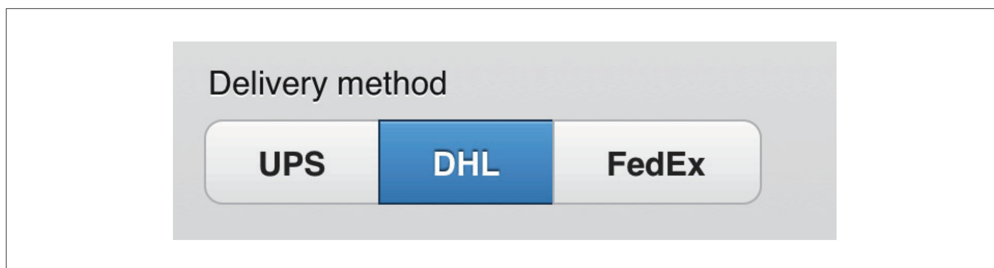


图 5-18：使用水平 **controlgroup** 容器时，单选按钮会被转换为开关按钮，选中的值将使用不同的色样显示



虽然 jQuery Mobile 显示的单选按钮的外观完全不同，但表单的工作方式是一样的。也就是说对应的 `value` 属性将与选中的元素对应。

5.2.11　复选框

复选框与单选按钮的工作方式类似（图 5-19），不过允许多选。下面是使用一个单独的复选框的例子：

```
<label for="accept">I accept terms and conditions</label>
<input type="checkbox" id="accept" name="accept" value="yes">
```

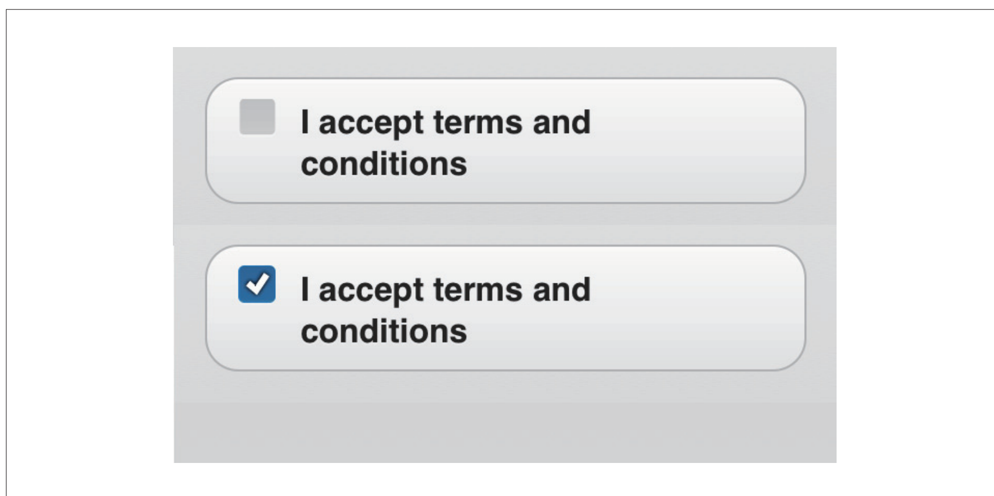


图 5-19：复选框与它对应的 **label** 被渲染为一个漂亮的按钮，选中时会显示一个复选框按钮

也可以把多个复选框放到一个 `controlgroup` 中，这样将得到一组多选按钮的界面（图 5-20）：

```
<legend>Delivery options</legend>

<div data-role="controlgroup">

<label for="optionGift">Pack it as a Gift</label>
<input type="checkbox" id="optionGift" name="optionGift" value="yes">

<label for="optionBag">Send it with a bag</label>
<input type="checkbox" id="optionBag" name="optionBag" value="yes">

<label for="optionRemove">Remove the box</label>
<input type="checkbox" id="optionRemove" name="optionRemove" value="yes">

</div>
```

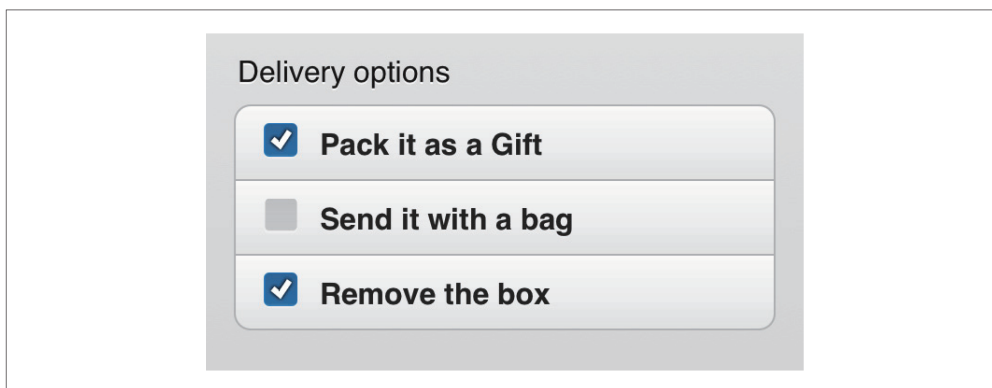


图 5-20：带 `legend` 元素的组合复选框创建了一个漂亮的多选列表

使用水平 controlgroup 时，界面将变成一个多选开关按钮控件，如图 5-21 所示。如果按钮被按下，表示它是一个被选中的元素。



图 5-21：如果复选框的文本标签较短，并且数量少于 5 个，则可以使用水平控件组来得到一个开关按钮形式的多选控件

```
<div data-role="controlgroup" data-type="horizontal">

<label for="bold">B</label>
<input type="checkbox" id="bold" name="bold" value="yes">

<label for="italic">I</label>
<input type="checkbox" id="italic" name="italic" value="yes">

<label for="underline">U</label>
<input type="checkbox" id="underline" name="underline" value="yes">

</div>
```

5.2.12 上传文件

由于在一些智能手机和平板电脑上缺少支持，如 iOS（iPhone 和 iPad）、Android 2.2 之前的版本以及 webOS，在移动设备上文件上传确实是个问题（图 5-22）。出于各种原因，这些平台不支持 `<input type="file">`，还缺少对用户公开的文件系统。jQuery Mobile 没有对上传文件提供特别的支持，在移动浏览器上实现文件上传时需要非常小心。

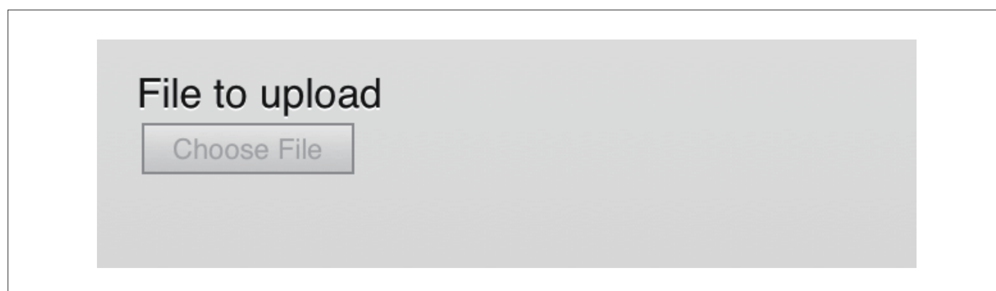


图 5-22：iOS 版 Safari 等部分移动浏览器及平台不支持文件上传，只能得到一个被禁用的标准表单控件

一些新的设备上，如 Android 3.0 版及更新版本的设备，开始支持 HTML 媒体捕获（HTML Media Capture）API，允许我们通过一个文件上传表单域请求摄像头图片、视频或音频。可以在 http://mobilehtml5.org 查看这个 HTML5 API 的兼容情况。

第6章

jQuery Mobile API

jQuery Mobile 提供了使用 JavaScript 与框架通信以及进行内容管理的 API。首先要知道的是它是一个 HTML5 框架，创建内容的最佳方式是使用无侵入性的 HTML5。

使用 JavaScript 代替标记语言来创建页面和内容时，在一些 B 级浏览器以及不支持 jQuery Mobile 的老平台上会遇到兼容性问题。如果你只关注现代平板电脑或智能手机，并且准备在不同的设备上测试你的代码，那么可以安全地使用 JavaScript 以及 AJAX 代替标记语言来创建内容。

这些 JavaScript API 不仅可用于创建与框架兼容的动态内容，还提供了新的事件机制以及可以自行定义的全局配置。

本章需要读者对 JavaScript 以及 jQuery 核心框架有一个基本的了解。

6.1 文档事件

网页上经常使用页面的 `load` 事件来配置默认值和初始化代码。使用 jQuery 核心框架时，你可能也喜欢 `document` 元素的 `ready` 事件。

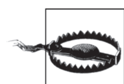
操作 jQuery Mobile 文档时，需要理解并处理一个新的事件：`mobileinit`。这个新事件在 jQuery Mobile 框架载入完成并已准备好执行初始化代码时触发。这个事件应该使用 jQuery 的 `bind` 方法，在 `document` 元素上处理：

```
$(document).bind('mobileinit', function(){
    // 这儿是初始化代码
});
```

mobileinit 事件会在 jQuery Mobile 框架载入内存之后、UI 元素被渲染之前触发。所以我们可以使用这个事件处理程序改变一些 UI 全局选项。

jQuery Mobile 文档事件的执行顺序通常是：

- mobileinit
- ready
- load



如果想在页面载入或显示之后再执行某些代码，则不应该使用 load、ready 或 mobileinit。每个 jQuery Mobile 页面元素都有一系列的可以绑定的事件。

关于 mobileinit 事件的第一个问题是应该把它放在哪儿。我们应该在 header 元素中的指定位置绑定这个事件，具体来说，就是在引入 jQuery 核心以及引入 jQuery Mobile 之间的位置。因为我们需要 jQuery 对象 $，并且需要在 jQuery Mobile 框架执行之前绑定它。

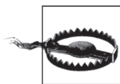
一个典型的带脚本配置代码的 jQuery Mobile 文档模板是这样的：

```
<!DOCTYPE html>
<html>
<head>
       <meta charset="utf-8" />
       <title>My first jQuery Mobile code</title>
       <link rel="stylesheet" href="http://code.jquery.com/mobile/1.0/
           jquery.mobile-1.0.min.css" />
       <script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>

       <!-- 自定义初始化代码 -->
       <script src="customcode.js"></script>

       <script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.
           js"></script>
     <meta name="viewport" content="width=device-width, initial-scale=1">
 </head>
</html>
```

记住，出于性能考虑，在一个移动页面上添加太多的 script 标签不是好习惯。有时候，将所有自定义的初始化代码放在 HTML 文档的内联脚本中比引入一个外部文件更好。



处理外部文档时需要注意，当用户通过首页访问你的网站时，那些通过 AJAX 加载的外部文档中的 script 标签将不会执行，所以所有东西都应该在一个外部 JavaScript 文件中定义好并且在每个文档中引入。

例如，可以像下面这样绑定 mobileinit 事件：

```
<script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>

<script>
$(document).bind("mobileinit", function() {
        // 这儿是初始化代码
});
</script>

<script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js">
    </script>
```

6.2 配置

jQuery Mobile 在 jQuery 主对象 $（或者 jQuery）上添加了一个新的 mobile 对象，因此，这个 API 的大部分工作都将使用 $.mobile（或者 jQuery.mobile）来完成。在使用 jQuery Mobile 时，你将发现很多全局属性以及有用的方法。这个对象只在 mobileinit 事件触发后才可用。

这个框架使用 jQuery UI 桌面版框架的部件结构。所谓部件即是由框架管理的控件。在 jQuery Mobile 1.0 中，部件通常通过 data-role 属性来映射，当然，也有不带 role 属性的表单控件。因此，page、button 以及 listview 都是框架内置的部件。

每一个部件都有一个对象构造器以及默认配置，这些默认配置可以在 mobileinit 中更改，更改后会影响到该页面上的每一个部件。

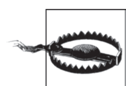
下面是 jQuery Mobile 1.0 中可用部件的列表：

- page
- dialog
- collapsible
- fieldcontain
- navbar
- listview
- checkboxradio
- button

- slider
- textinput
- selectmenu
- controlgroup

一些 jQuery Mobile 富控件被组织为一个部件。例如，所有文本输入类型（包括 textarea）都使用同一个部件 textinput，复选与单选按钮则都属于 checkboxradio 部件。

每个部件都有它自己的对象构造器，这些构造器代表了页面上每个对象的工作方式。可以通过 $.mobile.<widget_name>.prototype 来访问相应的原型。通常，每个部件构造器都有一个 option 对象，通过这个对象可以定义部件的默认属性，例如，$.mobile.page.prototype.options 可用于定义应用到每个页面实例（data-role="page"）的默认属性。



记住，要在 mobileinit 事件处理程序中设置全局或部件的默认属性，如果在这个事件之前或之后改变默认属性，则新的属性可能不会在文档中生效。

6.2.1 全局配置

我们可以使用 data-* 属性在各个元素上显式定义的大部分值，都可以全局定义并应用到每个控件上，除非标签中定义了新的值。

大多数全局配置的值都可以在 mobileinit 事件处理程序中通过 $.mobile 对象修改。

1. 用户界面

默认情况下，jQuery Mobile 会为页面上的特定元素分配一些类，当它们被激活时再分配别的类。这些类根据主题样式表提供不同的外观。通过修改字符串值的属性 activePageClass 和 activeBtnClass 可以修改当前活动页面（在多页面的文档中）或激活按钮的类名。默认情况下，它们的值分别为 ui-page-active 和 ui-btn-active。很多其他部件上也会使用激活按钮状态，如基于按钮的导航条、单选按钮以及复选按钮。

有两个（大多数情况下我们不会修改的）全局属性可用于修改滚动行为。默认情况下，jQuery Mobile 文档加载后，它会从顶部向下滚动到恰好隐藏掉地址栏的位置。可以通过设置 defaultHomeScroll 的值来改变这个行为。当打开一个页面然

后再一次回到这个页面时，框架会将视口滚动到之前点击时该页面所在的位置（记住位置）。不过，如果该页面非常接近顶部（但不是在0像素的位置），框架则将停留在顶部而不会滚动。返回页面时默认的最小滚动值是250像素高，可以通过minScrollBack属性修改这个值。

最后，两个我们会经常改动的全局属性是页面和对话框加载的默认过渡效果。默认情况下，它们的值为slide以及pop，可以通过defaultPageTransition和defaultDialogTransition属性来改变它们。

下面的示例将改变若干用户界面的默认值：

```
$(document).bind("mobileinit", function() {
        // 改变默认值
        $.mobile.defaultPageTransition = "fade";
        $.mobile.minScrollBack = 150;
        $.mobile.activeBtnClass = "active-button";
});
```

2. 核心及AJAX功能

jQuery Mobile的一些核心功能可以通过对应的全局属性来操作。如果我们在使用另一个可能会与jQuery Mobile冲突的框架，可以使用ns全局属性来定义一个命名空间。默认情况下命名空间没有定义，定义命名空间的示例代码如：

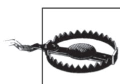
```
$(document).bind("mobileinit", function() {
        // 改变默认值
        $.mobile.ns = "firt";
});
```

之后，所有data-*属性将变成data-<命名空间>-*，如上面的例子中将变成data-firt-*，这包括每一个jQuery Mobile自定义属性如data-role（将转变为data-<命名空间>-role）。在新的命名空间下，典型的页面模板将变成：

```
<div data-firt-role="page">
   <div data-firt-role="header" data-firt-theme="a">

   </div>
   <div data-firt-role="content">

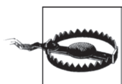
   </div>
</div>
```



如果定义了命名空间，那么也需要手动修改对应的CSS文件（包括结构和主题文件）以便识别新命名空间。例如，需要将[data-role=page]条件选择符替换为匹配新命名空间的值，如[data-命名空间-role=page]。

jQuery Mobile 的核心功能之一是用于加载外部页面的 AJAX 框架。我们可以设置布尔值属性 ajaxEnabled 来禁用所有的 AJAX 功能。如果使用 $.mobild.ajaxEnaled=false 把它禁用了，那么所有外部页面都将通过浏览器使用完整的 HTTP 请求加载。

默认情况下，XMLHttpRequest（AJAX 背后的对象）不支持跨域请求。这意味着，如果我们的页面在 domain1.com 上，那就不能通过 AJAX 加载 domain2.com 的页面。这种情况下，框架将自动使用完整的 HTTP 请求。在一些特殊情况下我们可以使用跨域请求，这时可以通过 allowCrossDomainPages 属性来强制框架支持。



一些新的移动浏览器支持跨域资源共享（Corss Origin Resource Sharing, CORS），这是一份 W3C 的草案，见 http://w3.org/TR/cors。根据这个标准，如果服务器返回了相同的指定的 HTTP 头，则浏览器将支持跨域请求。可以在 http://mobilehtml5.org 上验证移动浏览器对 CORS 的支持。

如果你正在创建一个离线应用或混合解决方案，如 PhoneGap 或 RhoMobile 应用，将主要使用 file:// 协议来加载页面（本地文件）。这些框架允许你发起对互联网上任何域名的 AJAX 请求，所以如果你正在创建这种应用并想通过网络加载外部页面，则应当使用 $.mobile.allowCrossDomainPages=true 来开启它。

有些部件，如嵌套列表视图，会动态地生成新的页面。每个新页面都需要有一个名字（用于 hash、URL 或其他用途）。默认情况下，jQuery Mobile 使用 ui-page 作为参数名，也可以通过 $.mobile 对象的 subPageUrlKey 属性改变这个值。对对话框来说，有一个默认与 ui-state=dialog 关联的 dialogHashKey 属性。通常情况下不需要修改这些属性。

框架默认会改变所有链接的行为，可以通过 $.mobile.linkBindingEnabled=false 来禁用这个特性，以便让 jQuery Mobile 支持不同的功能。默认情况下，jQuery Mobile 会在 DOM 就绪后对文档中的第一个页面执行初始化，可以通过 $.mobile.autoInitializePage=false 来禁用这个行为。

通过 $.mobile.hashListeningEnabled=false，我们也可以禁用自动散列读取，即禁止用户在点击浏览器或设备的前进和后退按钮时页面自动前进和后退的行为。

3. 本地化字符串

jQuery Mobile 中有一些硬编码的字符串值，可以将它们修改或本地化为其他语言。其中一些值在典型的 jQuery Moile 文档中不可见，这是因为它们是用于语义信息或可访问性的（以便屏幕阅读器知道它们是什么）。

可修改的字符串包括：使用 AJAX 加载外部页面时显示的加载消息、外部页面无法加载时的加载错误消息以及一些其他基于组件的消息。

下面是各个消息的列表及默认值：

```
// 全局字符串
$.mobile.loadingMessage = "loading";
$.mobile.pageLoadErrorMessage = "Error Loading Page";

// 组件字符串
$.mobile.page.prototype.options.backBtnText = "Back";
$.mobile.dialog.prototype.options.closeBtnText = "Close"
$.mobile.collapsible.prototype.options.expandCueText = " click to expand
contents";
$.mobile.collapsible.prototype.options.collapseCueText = " click to collapse
contents";
$.mobile.listview.prototype.options.filterPlaceholder = "Filter items...";
$.mobile.selectmenu.prototype.options.closeText = "Close";
```

因此，可以使用类似下面的代码来创建一个中文版本的 jQuery Mobile 文本方案：

```
$(document).bind('mobileinit', function() {
  // 全局字符串
  $.mobile.loadingMessage = "正在加载";
  $.mobile.pageLoadErrorMessage = "加载页面出错;

  // 组件字符串
  $.mobile.page.prototype.options.backBtnText = "后退";
  $.mobile.dialog.prototype.options.closeBtnText = "关闭"
  $.mobile.collapsible.prototype.options.expandCueText = " 点击展开";
  $.mobile.collapsible.prototype.options.collapseCueText = " 点击收起";
  $.mobile.listview.prototype.options.filterPlaceholder = "过滤";
  $.mobile.selectmenu.prototype.options.closeText = "关闭";

});
```

4. Touch overflow

前面我们讨论过，在 jQuery Mobile 1.0 中，固定工具栏（使用 `data-position="fixed"`）并不会生成一个真正固定的工具栏，只会在不滚动页面的情况下模拟固定工具栏。如图 6-1 所示。

在 iOS 5.0 中，Safari 支持 `position:fixed` 元素，使用 `overflow:scroll` 可支持在区块中单指滚动，还支持一个新的名为 `overflow-scrolling:touch` 的预定义扩展。默认情况下，这个行为是被禁用的，不过如果想在 iOS 5（将来可能还包括其他平台）中提供一个真正的固定工具栏，只需启用 `touchOverflowEnabled` 即可。在不支持的平台上，它会降级为普通的 jQuery Mobile 固定工具条。

```
$(document).bind('mobileinit', function() {
    $.mobile.touchOverflowEnabled=true;
});
```

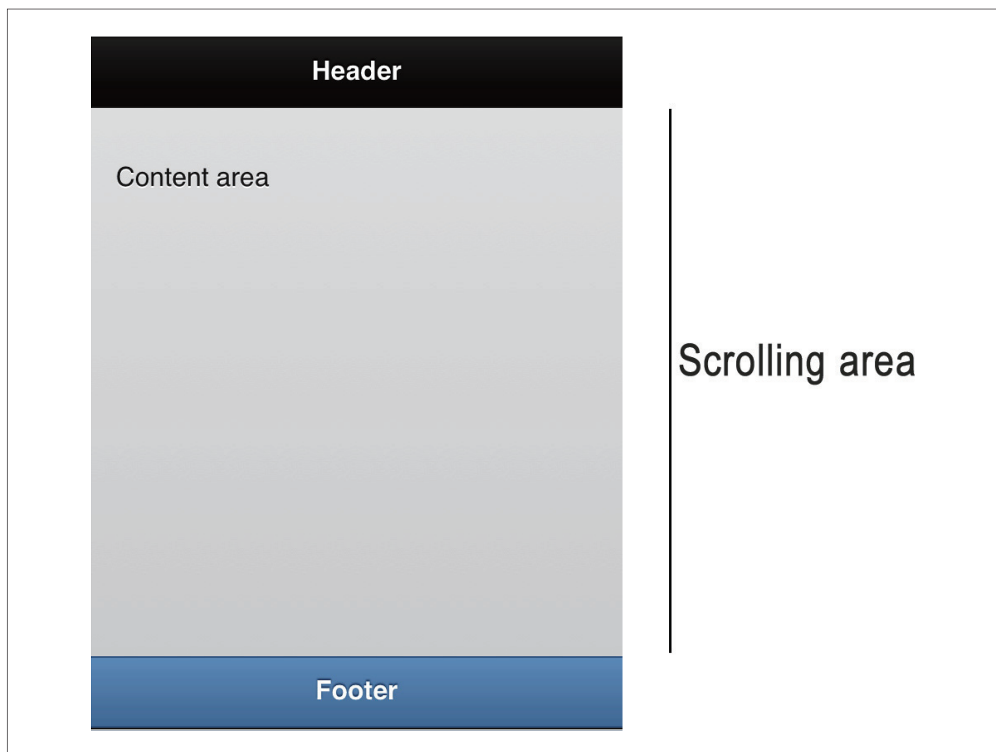


图 6-1：作用触摸溢出固定页头后，页面将以另一种方式设计——内容区域将有自己的滚动区



Android 3.0 及 PlayBook 上的 BlackBerry 浏览器也支持使用 `overflow: scroll` 来实现块级元素内的滚动。jQuery Mobile 1.1 版开始将自动支持这些浏览器。

开启这个属性之后，还可以使用 `touchOverflowZoomEnabled` 来开启缩放。这个选项可能会引起可用性问题，因此请小心使用。未来这两个属性将不能同时开启，因为它们将被默认的固定工具栏的行为替代。

6.2.2　页面配置

页面由 `data-role="page"` 定义，每个页面都有若干默认选项，这些默认选项可以通过 `data-*` 属性来修改。要改变默认设置，只需改变每个页面实例的 `prototype` 的 `options` 属性即可。

例如，如果想为每个访问历史记录中包含前一页的页面提供一个可见的后退按钮，只需指定 addBackBtn 属性为 true。还可以通过 backBtnText 和 backBtnTheme 来改变后退按钮的文本及主题。

除此之外，也可以通过指定 headerTheme、footerTheme 以及 contentTheme 来改变默认的主题。

例如：

```
$(document).bind('mobileinit', function() {
  $.mobile.page.prototype.options.addBackBtn = true;
  $.mobile.page.prototype.options.backBtnTheme = "e";
  $.mobile.page.prototype.options.headerTheme = "b";
  $.mobile.page.prototype.options.footerTheme = "d";
});
```



创建全屏 WebApp、混合或 PhoneGap 应用时，应该总是在页头中提供一个可见的后退按钮，为了实现这一点，需要处理 mobileinit 事件，并指定 $.mobile.page.prototype.options.addBackBtn=true。

页面加载

每次通过 AJAX 从外部加载页面时，都有一些默认属性被应用。可以从 $.mobile.loadPage.defaults 对象找到这些属性，表 6-1 列出了它们可能的值。

表6-1：$.mobile.loadPage.defaults属性

属　　性	可接受的值	默　认　值	描　　述
type	"get"/"post"	"get"	指定 AJAX 请求类型
data	object/string		如果 type 为 "post"，可以这儿设置要发送的数据
reloadPage	trye/false	false	指定如果页面已在 DOM 中有缓存的情况下是否重新加载页面
role	string	< 由 data-role 指定 >	指定应用到目标页面的 role
showLoadMsg	true/false	true	指定请求超时的情况下是否显示加载消息
loadMsgDelay	milliseconds	50	在显示加载消息之前等待多少毫秒
theme	a 到 z	c	默认应用到各页面的色样
domCache	true/false	false	指定是否将页面缓存在 DOM 中



如果只是想改变一个页面而不是所有页面的加载属性，可以使用 $.mobile.changePage 属性。

6.2.3 部件配置

jQuery Mobile 中的每个部件（widget）都有自己的默认配置属性。记住，可以通过部件的 prototype 属性中的 options 对象来改变部件的默认设置，接口为 $.mobile.< 部件名 >.prototype.options。例如：

```
$(document).bind('mobileinit', function() {
    // 在所有 listviews 中启用过滤
    $.mobile.listview.prototype.filter = true;

    // 在所有选择框上启用非原生菜单
    $.mobile.selectmenu.prototype.nativeMenu=false;
});
```



大多数部件都支持 theme 默认属性。因此，如果想让所有选择菜单都默认使用主题 e，可以使用 $.mobile.selectmenu.prototype.theme="e" 来实现。

表 6-2 展示了我们可以改变的部件属性。

表6-2：各部件默认属性

部件	属性	值（默认）	描述
< 所有部件 >	theme	a 到 z	应用到每个部件实例的色样
listview	filter	true/(false)	在每个列表视图上开启过滤
listview	filterPlaceHolder	string	过滤文本框中的占位文本
listview	filterTheme	a 到 z	过滤文本框的色样
navbar	iconpos	(top)/bottom/left/right	导航栏元素的图标位置
slider	trackTheme	a 到 z	滑动开关的色样
selectmenu	icon	icon 值 (arrow_d)	选择菜单打开按钮的图标
selectmenu	iconpos	top/bottom/left/(right)	图标位置
selectmenu	corners	(true)/false	打开按钮使用圆角
selectmenu	shadow	(true)/false	打开按钮使用阴影
selectmenu	iconshadow	(true)/false	打开按钮的图标使用阴影
selectmenu	menuPageTheme	a 到 z(b)	导航菜单的色样
selectmenu	overlayTheme	a 到 z(a)	非原生覆盖层的色样
selectmenu	closeText	string	非原生菜单的关闭按钮的文案
selectmenu	nativeMenu	(true)/false	指定是否使用原生选择菜单
dialog	closeBtnText	string	关闭按钮的文案
dialog	overlayTheme	a 到 z(a)	对话覆盖层的色样
collapsible	expandCueText	string	打开按钮的文案
collapsible	collapseCueText	string	关闭按钮的文案
collapsible	collapsed	(true)/false	指定当前折叠部件默认为开启还是关闭
collapsible	heading	英文逗号分隔的值	用于折叠部件头部的标签列表
collapsible	contentTheme	a 到 z	内容的色样

（续）

部　　件	属　　性	值（默认）	描　　述
collapsible	iconTheme	a 到 z	页头图标的色样
button	icon	图标值	每个按钮的默认图标
button	iconpos	图标位置	每个按钮图标的默认位置
button	inline	true/(false)	每个按钮是否为内联
button	corners	(true)/false	每个按钮是否使用圆角
button	shadow	(true)/false	每个按钮是否使用阴影
button	iconshadow	(true)/false	按钮图标是否使用阴影

6.3　实用工具

jQuery Mobile 提供了很多使用 JavaScript 来管理应用的实用工具，这些工具通过方法以及只读属性提供，让我们使用 JavaScript 创建更好的体验。

6.3.1　Data-*工具

使用 jQuery Mobile 时经常需要处理 data-* 自定义属性。例如，想取得页面上按钮的集合，可以使用 jQuery：

```
var buttons = $("a[data-role=button]");
```

jQuery Mobile 添加了一个新的名为 jqmData 的过滤器，并会应用我们指定的命名空间。上面的代码可以很容易并且安全地更改为：

```
var buttons = $("a:jqmData(role='button')");
```

同时，在 jQuery 集合对象上应该使用 jqmData 和 jqmRemoveData 来代替原来的 jQuery 函数 data 和 removeData，例如：

```
$("a").jqmRemoveData("transition");
$("#button1").jqmData("theme", "a");
```

6.3.2　页面工具

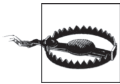
如果需要访问当前页面，可以使用 jQuery Mobile 提供的 $.mobile.activePage 属性，该属性自动与当前可见的 data-role="page" 元素关联。这个属性指向对应的 jQuery DOM 对象（通常是一个 div 元素）：

```
var currentPageId = $.mobile.activePage.id;
```

可以通过 $.mobile.pageContainer 属性访问当前页面的容器（通常为 body 元素）。

框架中最有用的工具是 $.mobile.changePage 方法，它允许我们转向另一个页面，就像用户点击了相应的链接一样。可以在 JavaScript 中通过这个方法来显示内

部或外部页面。



一些工具在 mobileinit 事件被触发后才可用，因此不能在这个事件处理程序中使用这些工具。

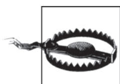
这个方法的第一个必选参数可以是一个字符串值（意味着它是一个外部页面的 URL），也可以是一个指向内部页面的 jQuery 对象。

可以像这样加载外部页面 external.html：

```
$.mobile.changePage("external.html");
```

要转向当前文档中已经存在的内部页面，可以使用：

```
$.mobile.changePage($("#pageId"));
```



不能只通过字符串 "#pageId" 来加载内部页面，必须通过 jQuery 传入对应的 DOM 对象，如 $("#pageId")。

页面过渡选项

changePage 方法的第二个可选参数是一个对象，通常为 JSON 格式，用于定义页面过渡以及 / 或者 AJAX 加载的可选选项。

所有选项见表 6-3。

表6-3：changePage操作的可选属性

属　　性	值	默　认　值	描　　述
transition	过渡名	slide	要应用的过渡效果
reverse	true/false	false	过渡效果是否反转（通常用于返回操作）
type	"get"/"post"	"get"	用于加载外部页面的 HTTP 方法
data	object 或 string		使用 POST 方法时发送的数据
allowSamePageTransition	true/false	false	允许转向当前激活的同一个页面（转向自身）
changeHash	true/false	true	决定新页面是否加入历史记录
data-url	string		将写入地址的 URL
pageContainer	jQuery DOM 对象		用于添加新页面的容器
reloadPage	true/false	false	强制刷新页面，即便它已经在当前 DOM 中缓存了
showLoadMsg	true/false	true	在若干毫秒后是否显示页面加载消息
role	page/dialog	由 data-role 定义	应用到新页面的 role

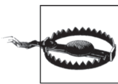
可以用类似下面的代码来强制使用反转 slide 过渡效果：

```
$.mobile.changePage($("page2"), {
    transition: "slide",
    reverse: true
});
```

下面的示例将通过 POST 方式发送数据并加载一个外部页面：

```
<script>
function viewProduct(idProduct) {
   $.mobile.changePage("productdetail.php", {
       method: "post",
       data: {
           action: 'getProduct',
           id: idProduct
       },
       transition: "fade"
   });
}
</script>

<!-- ... -->
<a href="javascript:viewProduct(5200)" data-role="button">Product details</a>
```



记住，即使通过 POST 方式加载页面，目标页面仍然必须是一个 jQuery Mobile 文档，包括页头以及 data-role="page"。

还有一个 $.mobile.loadPage 方法，主要由 changePage 在加载外部页面时使用。这个方法会将目标页面加载到当前 DOM 中，但并不会转向它。要转向新页面，需要使用 changePage。可以使用 loadPage 预加载指定内容并将它插入 DOM 中，然后再使用对应的 jQuery DOM 对象通过 changePage 方法转向它。

6.3.3 平台工具

框架为我们提供了一些平台工具，可用于协助网站开发。表 6-4 显示了最有用的平台工具。

表6-4：$.mobile对象中的用于查询当前平台的工具

方法/属性	描　　述
orientationChangeEnable	原生的 orientationchange 事件是否可用
gradeA()	如果当前浏览器在 jQuery Mobile 的兼容表中是 A 级，则返回 true
urlHistory	页面不刷新的情况下在 jQuery Mobile 中访问过的页面的集合，每个元素都有 pageUrl、title、以及 transition 属性
getDocumentUrl()	返回原始文档的 URL（第一个加载的文档）

（续）

方法/属性	描　　述
`getDocumentBase()`	返回原始文档的 base 属性
`keyCode`	用于处理键盘事件的预定义的常量，包括：ALT、BACKSPACE、COMMAND、COMMAND_LEFT、COMMAND_RIGHT、DELETE、DOWN、UP、RIGHT、LEFT、END、ENTER、ESCAPE、HOME、INSERT、MENU、PAGE_DOWN、PAGE_UP、PERIOD、SHIFT、SPACE、TAB、WINDOWS
`getScreenHeight()`	取得当前屏幕的高度

6.3.4　路径工具

jQuery Mobile 中有一些路径管理工具，可以通过公共方法 `$.mobile.path` 访问。具体方法见表 6-5。

表6-5：`$.mobile.path`对象中的路径工具

方　　法	描　　述
`parseUrl(url)`	返回一个对象，各个属性对应该 URL 的各个部分（protocol、hostname、port、pathname、directory、filename、hash 以及更多）
`makePathAbsolute(relativePath, absolutePate)`	基于相对路径返回绝对路径
`makeUrlAbsolute(relativeUrl, absoluteUrl)`	基于相对 URL 返回绝对 URL
`isSameDomain(Url1, Url2)`	如果两个 URL 同域名则返回 `true`
`isRelativeUrl(Url)`	如果 URL 是相对地址则返回 `true`
`isAbsolute(Url)`	如果 URL 是绝对地址则返回 `true`

6.3.5　UI工具

最后一类工具是关于用户界面的。使用 `$.mobile.getInheritedTheme(element, defaultSwatch)` 方法，可以取得元素基于色样定义或继承链应该应用的色样。

使用 `$.mobile.silentScroll(y)` 可以将页面滚动到任意位置，同时不显示动画，也不触发任何事件。使用 `$.mobile.showPageLoadingMsg()` 和 `$.mobile.hidePageLoadingMsg()`，可以显示及隐藏弹出的加载消息。

```
// 这段代码将显示加载消息，并在 2 秒后将其隐藏
$.mobile.showPageLoadingMsg();
setTimeout(function() {
    $.mobile.hidePageLoadingMsg();
}, 2000);
```

最后，可以通过 `$.mobile.fixedToolbars.show()` 以及 `$.mobile.fixedToolbars.hide()` 显示及隐藏固定工具栏（如前几章所讲述的那样）。工具栏可以全屏或者固

定显示。在 iOS 5 中，无法隐藏真正的固定工具条。默认情况下，工具条显示或隐藏时使用淡入淡出效果，可以指定一个 true 参数，$.mobile.fixedToolbars.show(true) 将立即显示工具栏，而不再会有淡入淡出动画。

6.4 自定义过渡

第三章提到过在 jQuery Mobile 1.0 中所有可用的过渡效果。我们可以定义自己的过渡效果吗？当然可以，并且有两种实现方法：

- 使用 CSS3 动画；
- 使用 JavaScript。

定义一个 data-transition 属性（或者从 JavaScript 中应用一个过渡）时，jQuery Mobile 首先会检查这个名字在框架中是否已经存在。如果不存在，则继续在 $.mobile.transitionHandlers 集合中查找。如果过渡名未知并且在集合中也未定义，则使用默认过渡效果。

默认的过渡效果可以被修改，它必须是一个处理函数。默认情况下，后备过渡被映射为一个使用 CSS3 动画（参见下一章）的自定义过渡。

也可以将默认过渡映射到 $.mobile.noneTransitionHandler，后者将简单地显示新页面并隐藏老页面，没有任何动画效果。

例如，可以添加一个 explode 过渡类型：

```
$.mobile.transitionHandlers.explode = explodeTransitionHandler;
```

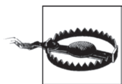
也可以修改默认的过渡，将它赋值为任何其他处理函数：

```
$.mobile.defaultTransitionHandler = explodeTransitionHandler;
```

过渡处理函数是一个 JavaScript 函数，接受四个参数：

- 过渡名；
- reverse，如果设置为 true，表示本过渡将反转调用；
- toPage，指向目标页面的 jQuery DOM 对象；
- fromPage，指向原始页面的 jQuery DOM 对象。

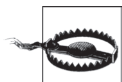
可以使用任意 JavaScript 代码来实现过渡效果，只需确认一点：最后需要从原始页面上移除 $.mobile.activePageClass，并将它应用到目标页面上。



使用基于 JavaScript 的过渡时要小心，在一些平台上可能会遇到兼容或性能问题。使用之前请先做大量的测试。

6.5 动态内容

使用数据库驱动站点等动态内容时，我们可能不想动态地创建那些 jQuery Mobile 文档，而想用 JavaScript 和 AJAX 来改变、显示以及隐藏 Web 应用中的信息。



在不支持 jQuery Mobile 的浏览器上，使用基于 JavaScript 的元素而不是语义标记时可能会遇到问题。不过如果目标为智能手机及平板电脑则不会有问题。

6.5.1 创建页面

可以动态创建页面吗？我们知道页面只是一个带有属性 `data-role` 的 `div` 元素，所以首先可以考虑通过这个特性实现。让我们试一下，创建一个基本的页面，再从这个基本的页面上使用 JavaScript 动态地创建四个页面：

```
<div data-role="page">
        <div data-role="header">
    <h1>Dynamic page</h1>
    </div>
        <div data-role="content">
        <a id="button1" href="javascript:addPages()" data-role="button">
            Add Pages</a>
        <ul id="list1">

        </ul>
    </div>
</div>
```

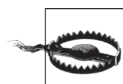
然后在一个脚本中我们定义动态创建页面的函数，并添加转向这些页面的按钮：

```
function addPages() {
  for (var i=1; i<5; i++) {
      var page = $("<div>").jqmData("role", "page").attr("id", "page" + i);
      // 页头
      $("<div>").attr("data-role", "header").append($("<h1>")
          .html("Page " + i)).appendTo(page);
      // 内容
      $("<div>").attr("data-role", "content").append($("<p>")
          .html("Contents for page " + i))
          .appendTo(page);

      $("body").append(page);
```

```
                $("<li>").append($("<a>").attr("href", "#page"+i).html("Go to
                    page " + i))
                .appendTo("#list1");
        }
            $("#button1").hide();
    };
```

图 6-2 显示了这个示例的实际效果，可以看到，在页面加载后动态创建的页面可以工作，但页头样式有误。



动态创建页面有一个缺点：如果用户在某个新创建的页面上刷新当前页面，必须监听 mobileinit 并检查（通过读取页面地址中的散列值或某个页面事件）用户是否正在加载某个动态页面，否则无法刷新。第二次加载时，动态页面将不存在，得根据需要再次创建。

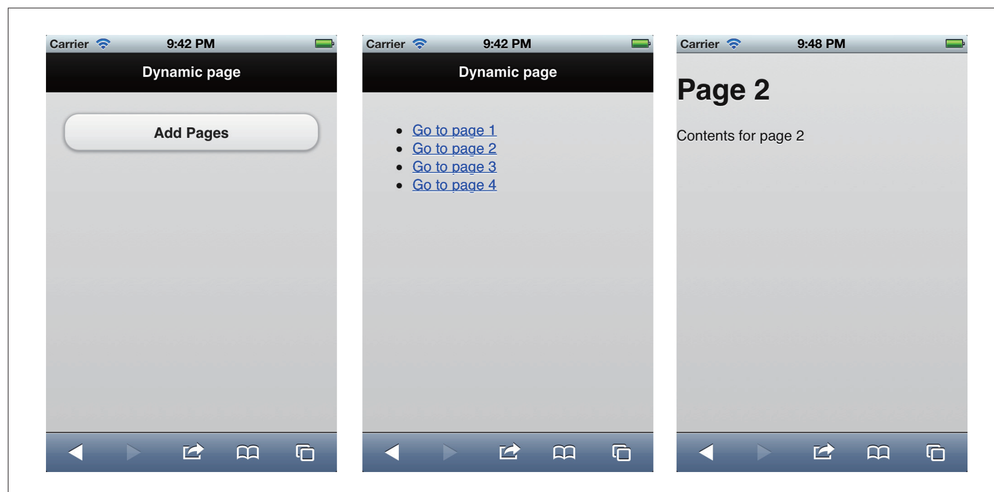


图 6-2：动态创建的页面与原来就有的页面一样可以工作

可以在 jQuery DOM 元素上调用 page() 方法来增强动态插入的页面，比如 $("#page1").page()。

创建动态页面的最佳方法是只连接它们，例如，连到 #page1 并捕获 pagebeforechange 事件，修改框架对应的行为。下一章将讲解这个事件，也可以通过下面的例子来理解它：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="UTF-8" />
<title>jQuery Mobile</title>
<script src="jquery.js"></script>
```

```
<link rel="stylesheet" type="text/css" href="jquery.mobile-1.0.css">
<script src="jquery.mobile-1.0.js"></script>
<script>
$(document).bind('pagebeforechange', function(event, data) {
        // 从 data.toPage 中收到目标页面，将它标准化
        var url = $.mobile.path.parseUrl(data.toPage).hash;

        if (url!=undefined && url.length>5 && url.substring(0, 5)=="#page") {
               // 动态插入一个新页面
               var id = url.substring(5);

               // 使用 DOM 中已经存在的页面模板
               $("#pageTemplate h1").html("Page " + id);

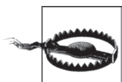
               // 转向真实的页面模板，同时不在访问历史上使用真实页面
               $.mobile.changePage($("#pageTemplate"), {dataUrl: data.toPage});

              // 阻止正常页面过渡
              event.preventDefault();
        }
});

</script>
<meta name="viewport" content="width=device-width,user-scalable=no">

</head>

<body>
<div data-role="page">
        <div data-role="header">
    <h1>Dynamic pages</h1>
    </div>
        <div data-role="content">
        <a id="button1" href="#page1" data-role="button">Page 1</a>
        <a id="button1" href="#page2" data-role="button">Page 2</a>
        <a id="button1" href="#page3" data-role="button">Page 3</a>
        <a id="button1" href="#page4" data-role="button">Page 4</a>
    </div>
</div>
<div data-role="page" id="pageTemplate">
  <div data-role="header">
    <h1>Header</h1>
  </div>
  <div data-role="content">Content</div>
  <div data-role="footer">
    <h4>Footer</h4>
  </div>
</div>
</body>
</html>
```



页面渲染完成后使用 JavaScript 插入其他组件时，如果不触发 create 事件，这些组件可能无法正常渲染。

6.5.2 创建部件

图 6-2 中有一个带四个项目的列表，但显然将 ID 为 `list1` 的列表转换成 `listview` 会更好一些。你可能在想：在对应的 `ul` 上加一个 `data-role="listview"`。不过这不会起作用，因为页面已经加载完成了，在初始化时这个列表没有被当作列表视图部件处理。

动态创建部件需要调用部件对应的构造器。每个部件都有自己的构造器，它们就是与部件名同名的 jQuery 函数，因此，如果我们执行 `$("#list1").listview()`，则该 `ul` 将马上被转换并渲染为一个列表视图（见图 6-3）。

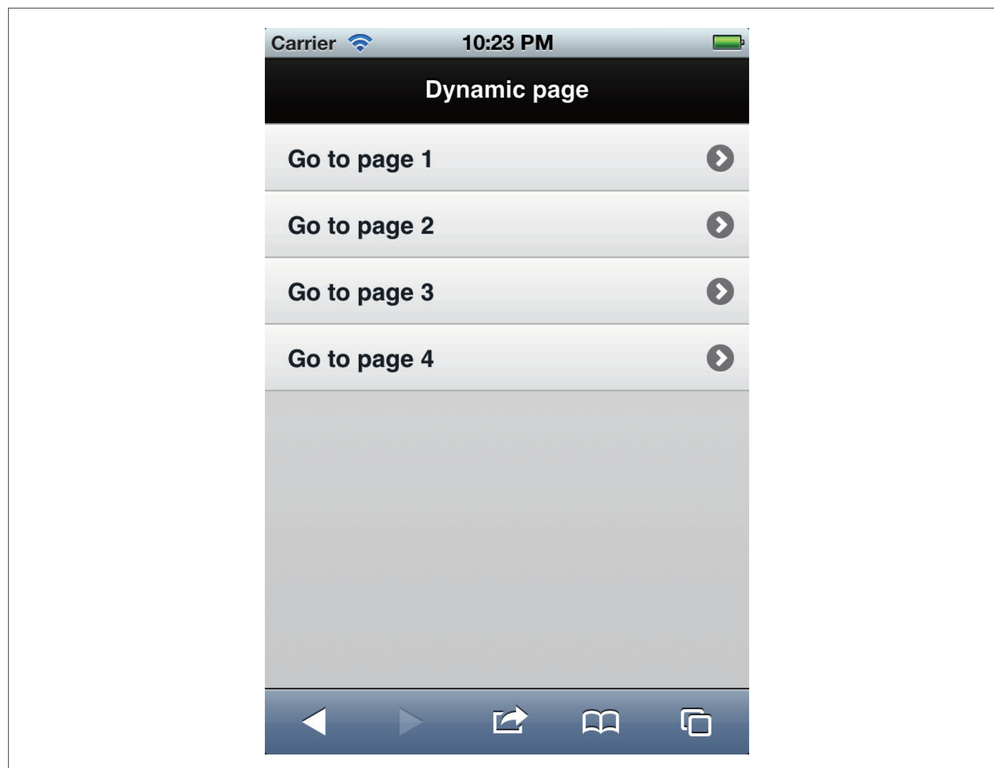


图 6-3：使用部件构造器可以动态创建任何部件，本例中创建的是列表视图

要将若干 `a` 元素转换为按钮，只需调用：

```
$("a").button();
```

也可以动态创建一个：

```
var button = $("<a>").attr("href", "somewhere.html").button();
```

6.5.3　更新部件

我们已经知道如何动态创建 jQuery Mobile 部件了，不过要修改一个已经被创建并渲染的部件的内容该怎么办呢？例如，往一个列表视图中添加元素，或改变一个复选框的值。

这种情况下需要刷新对应的部件，要实现这一点需要调用部件函数并传入字符串参数 `refresh`，例如：

```
$("#list1").listview("refresh");
$("#checkbox").val("true").checkboxradio("refresh");
```

回到最后一个例子，如果 `list1` 一开始被定义为一个空 `listview`，则我们需要在添加列表元素后刷新这个部件。否则，这个列表元素的界面将出现渲染错误：

```
<ul id="list1">
</ul>
```

添加元素后，可以调用：

```
$("#list1").listview("refresh");
```

6.6　创建网格

上一章曾提到有一个可用于创建 CSS 网格的特殊部件。要实现这个部件，需要使用一个带子元素的 HTML 元素，并根据其子元素的数目将它转换为 *n* 列的网格。

使用时只需调用 jQuery 函数 `grid`，例如：

```
$("#element").grid();
```

根据它的子元素的数目，框架将自动为该元素应用正确的 `ui-grid-<letter>` 类，为其子元素应用 `ui-block-<letter>` 类。

6.7　改变页面内容

如果改变了包含多个部件的一大块 HTML，例如使用通过 AJAX 请求收到的 JSON 数据创建了一个包含各种内容的 `collapsible` 元素，这时需要刷新整个容器。还有一个例子是向一个表单中添加了多个 `input` 元素，希望这些 `input` 被转换为部件，就像它们一开始就在页面中一样。

要刷新一个容器，只需在页面上触发 `create` 事件，这时每个部件就会再次检查是否需要创建新的实例。

例如：

```
$("#content").html(newHTMLcontentWithWidgets);
$("#page1").trigger("create");
```

通常每个部件的构造器都会响应页面的 `create` 事件，以便检查是否需要创建对应的控件。

6.8 处理事件

jQuery Mobile 提供了一些新的事件，可以使用 `bind` 或 `live` 等典型的 jQuery 方法来处理。

6.8.1 页面事件

普通 HTML 页面事件大家都已经很熟悉了，比如由浏览器应用到当前会话的针对每个 HTTP 页面加载的 `load` 及 `DOMready` 事件。jQuery Mobile 框架中有一些特别的元素可应用事件。就像我们已经知道的，jQuery Mobile 文档有不同的页面（内部或从外部加载），因此必须认真地考虑 jQuery Mobile 页面的加载。

每个页面（带 `data-role="page"` 的元素）都有一组不同的事件，这些事件有一些可以全局处理（同时处理所有页面），有一些则只对某个特定页面有效。

要全局处理页面事件，可以调用 `$(document).bind`，也可以调用更明确的 `$(":jqmData(role='page')").bind`。或者使用 `live` 来代替 `bind`，以便能绑定那些将来加入到 DOM 中的页面。

每个页面都有创建、加载中以及显示事件。

1. 创建事件

每个页面都有对应的创建或初始化事件，这些事件如下。

- `pagebeforecreate`
  页面已插入 DOM，但是组件还未创建。

- `pagecreate`
  页面已被创建，但组件还未渲染。

- `pageinit`
  页面已完全加载。这应该是页面最常用的事件。

- `pageremove`
  页面已从 DOM 移除（通常这是一个 AJAX 加载的页面并且当前未激活）。

例如，可以使用 jQuery 的 `live` 方法绑定一个 `pageinit` 事件：

```
$("#page2").live("pageinit", function(event) {

});
```



记住，使用 `bind` 绑定事件时，对应的元素必须在 DOM 中存在。如果对应元素不存在，可以使用 `live` 来代替。jQuery 1.7 支持一个新的 `on` 函数，但在 jQuery Mobile 1.0 及 jQuery 1.6.4 组合中还不可用。

2. 加载事件

不是所有页面都会默认与第一个 jQuery Mobile 文档一起加载，对那些使用 AJAX 加载的页面来说，事件处理程序通常绑定在 `$(document)` 上，因为此时对应的页面还不在 DOM 中，无法绑定对应处理程序。

可用的加载事件有以下这几个。

- `pagebeforeload`
  在所有 AJAX 请求完成之前执行。

- `pageload`
  当新页面已被加载并插入到 DOM 后执行。

- `pageloadfailed`
  指定页面无法加载时执行。

这些事件处理程序都会收到两个参数，一个事件对象和一个数据对象。

第一个参数是典型的事件处理程序值，具有阻止默认行为的 `preventDefault()` 方法。使用这个方法，可以强制框架不显示默认的错误消息警告并提供自己的 UI：

```
$(event).bind("pageloadfailed", function(event, data) {
     data.preventDefault();
     // 自定义错误管理
});
```

第二个参数是一个包含不同属性的对象，这些属性如下所示。

- `url`
  用于 `$.mobile.loadPage` 发起请求的绝对或相对 URL。

- absUrl
  绝对 URL。

- dataUrl
  用作页面标识的 URL。

- options
  所有传给 $.mobile.loadPage 的选项，例如告知是 GET 还是 POST 请求。

- xhr
  对应的 XMLHttpRequest 对象，用于更底层操作。

- textStatus
  用于错误消息。

- errorThrown
  错误异常对象，仅在 pageloadfailed 事件中有效。

- deferred
  仅在 pagebeforeload 和 pageloadfailed 事件中，并且调用了 event.preventDefault() 时有效。这些情况下，必须调用这个对象的 resolve() 或 reject() 方法告诉框架如何处理。



如果需要处理页面初始化，应该避免使用 load、ready 或 mobileinit 事件。正确的事件是 pageinit，它在每个页面上都有效。如果在 mobileinit 中绑定它，则应该使用 live 而不是 bind。

3. 显示事件

一个页面可以被初始化一次，但户可以在页面访问历史中前进及后退，因此，每个页面都可以被显示很多次。这就是为什么需要处理页面的显示及隐藏事件。

这些事件分成页面改变事件和过渡事件。

有效的页面变化事件包括以下这几个。

- pagebeforechange
  在页面改变发生之前以及过渡开始之前执行。

- pagechange
  在页面改变完成之后执行。

- `pagechangefailed`
  页面改变无法完成时执行。
  每个事件处理程序接受两个参数。

- `toPage`
  如果目标页面是外部页面，则值为目标页面的 URL 字符串；如果是内部页面，则值为目标页面的 DOM 对象。

- `options`
  与发送到 `$.mobile.changePage` 的选项相同。

可用的过渡事件有以下这几个。

- `pagebeforeshow`
  在过渡并显示页面之前执行（页面仍然处于隐藏状态）。

- `pageshow`
  在页面已完成加载过渡并正显示在屏幕上时执行。

- `pagebeforehide`
  在页面隐藏之前执行（页面仍然可见）。

- `pagehide`
  页面已完成卸载过渡并已隐藏时执行。

每个过渡事件的处理程序都能收到一个参数，值为相关页面的 jQuery（封装的 DOM）对象。如果是一个显示事件，对应的是上一页的对象；如果是隐藏事件，对应的则是下一页的对象。

6.8.2　部件事件

每个能动态显示或隐藏内容的部件，例如 `collapsible` 等，都会触发一个 `updatelayout` 事件，因为页面布局已经发生了改变。有时可能需要监听这个事件，以便更新 UI 上的其他内容。

6.8.3　方向事件

移动设备可以旋转，因此至少会有两个不同的方向：纵向和横向。有时我们想在方向改变时也改变应用的一些外观或行为。为此，jQuery Mobile 为我们提供了一个 `orientationchange` 事件，可附加在 `document` 上。

目前有些设备还没有原生支持 orientationchange 事件，在这些设备上 jQuery Mobile 会将它与 resize 事件关联。一些平台上，orientationchange 事件被触发时，窗口框架仍然是原来的状态，此时将无法取得正确的宽度及高度值。如果想强制仅当宽度和高度值更新后才触发这个事件，可以执行代码 $.mobile.orientationChangeEnabled=false。

这个事件对应的处理程序将收到一个字符串作为第一个参数，值为 portrait（表示纵向）或 landscape（表示横向）。无论在什么平台上，这两个值都是正确的（没有上面提到的宽度或高度的问题）：

```
$(document).bind("orientationchange", function(orientation) {
    if (orientation=="landscape") {
         // 现在是纵向
    } else {
         // 现在是横向
    }
});
```

6.8.4 手势事件

jQuery Mobile 提供了一些可绑定在任何 DOM 元素上的手势触摸事件。jQuery Mobile 1.0 中提供的手势事件有以下几个。

- tap
  在屏幕上快速地触摸一下时触发。
- taphold
  用户触摸屏幕并持续按住一秒钟时触发。在显示上下文菜单时很有用。
- swipeleft
  用户的手指从右划到左时触发。
- swiperight
  用户的手指从左划到右时触发。

下面的例子将把页面的 swiperight 事件绑定到返回处理函数：

```
$(document).bind("mobileinit", function() {
   $("#page2").live("swiperight", goBackToPage1);
});

function goBackToPage1() {
   $.mobile.changePage("#page1", { reverse: true });
   $("#page2").unbind("swiperight", goBackToPage1);
}
```

6.8.5 虚拟点击事件

虚拟点击？听起来是不是很奇怪？让我们先简单地讲解一下。大多数移动浏览器中，使用点击事件（如 `click`、`mouseover`）时都有 300-500 毫秒的延迟，但在使用触摸事件（如 `touchstart`、`touchmove`）时就没有这些延迟。另一个问题是并非所有触摸浏览器都支持触摸事件。

虚拟点击事件是一个包装，可用于取代触摸或点击事件，基于运行平台的不同它们将自动选择正确的事件。它还标准化了位置信息，只可用于单点触摸（非多点触摸）。

虚拟点击事件的用法与点击事件完全相同，不过名称中有一个 `v` 作为前缀。jQuery Mobile 中包括以下虚拟点击事件：`vclick`、`vmousedown`、`vmousemove`、`vmouseup` 和 `vmousecancel`。

第 7 章

创建主题

jQuery Mobile 允许开发者通过调整主题和 CSS 来定制整套用户界面。因为 jQuery Mobile 生成的是 HTML 和 CSS，所以，它的每一个样式都能用 CSS 来覆盖。

图 7-1 展示了多个不同主题的 jQuery Mobile 站点。如果想要浏览更多自定义界面的站点主题，不妨看看 http://jqmgallery.com。



图 7-1：jqmgallery.com 展示了众多使用 jQuery Mobile 创建的 Web 应用，其中不乏一些与标准主题相比已经迥然不同了的漂亮界面

主题是一组配色方案，包括：

- 文字颜色；
- 背景颜色和渐变；
- 字体。

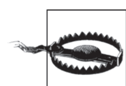
jQuery Mobile 支持从字母 a 到 z 至多 26 个色样定义。通常，我们至少会定义 5 个不同的色样。

每个主题都包含一个全局定义，适用于主题内的所有色样，它可以定义如下内容：

- 文本效果和盒效果，例如阴影和圆角；
- 按钮和其他控件的激活状态。

使用全局定义可以确保无论主题套用哪个色样都能保持一致的用户体验。比如，在默认主题里，无论按钮在常态下被设置为什么颜色，它的选中状态始终显示为蓝色。

主题保存在一个 CSS 文件中，这个 CSS 文件将与另一个由框架提供的描述文档结构的 CSS 文件一块被包含到 HTML 页面中。为避免和后续版本产生冲突，最好不要修改框架提供的结构性 CSS 文件。如果确实想要覆盖其中的样式，建议另写一个 CSS 文件，然后在载入结构性 CSS 文件之后再引入它（从而实现样式的覆盖）。



主题不应该包括关于元素大小和位置信息的定义。这类信息应该由结构性 CSS 来描述，而且，除非十分清楚自己在干什么，否则不要轻易修改这个结构性 CSS 文件。

7.1 ThemeRoller

创建主题最简单的方式莫过于使用免费的在线工具 ThemeRoller http://jquerymobile.com/themeroller。 用户可以使用右侧的审查器（Inspector）面板定义页面上每个元素的颜色（图 7-2），此外，把颜色直接拖到元素上也可以达到相同的效果。

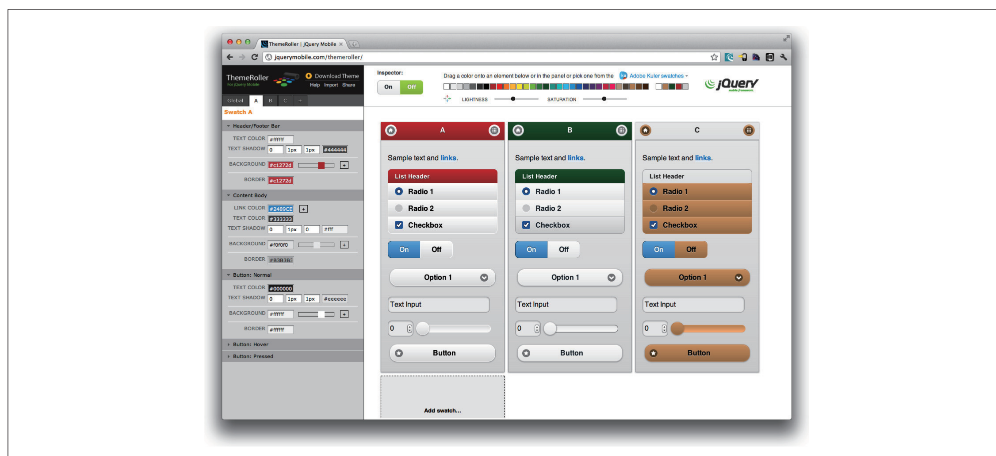


图 7-2：通过桌面浏览器访问 ThemeRoller

ThemeRoller 的主窗口分成三个窗格：

- 左侧的色样选择器；
- 顶部的调色板窗格；
- 右侧的预览窗格。

从顶部调色板选中一种颜色，拖放到预览窗口中某个元素的背景或者某段文字上，目标元素就会自动套用选中的颜色。



可以通过调节亮度和饱和度滑动条从调色板中获得更多的颜色。

7.1.1 全局设置

在左侧的全局设定标签页 (图 7-3) 里，可以定义：

- 字体；
- 按钮和其他控件的激活状态；
- 按钮和其他控件组的圆角半径；
- 图标属性；
- 盒阴影。

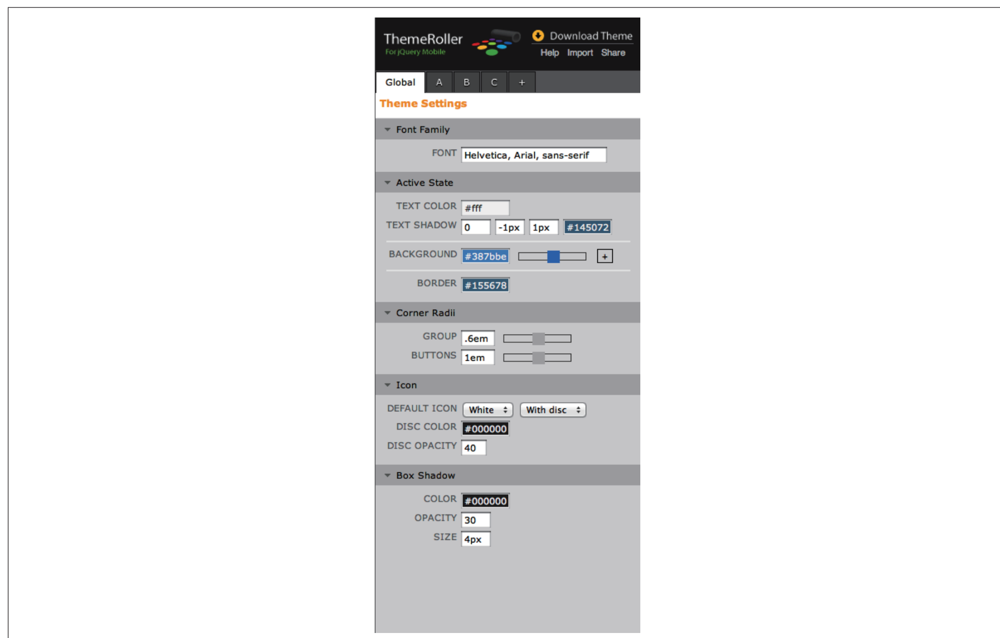


图 7-3：全局设置标签页

7.1.2 色样设置

从全局设置标签页可以切换到色样设置标签页，选择一个字母（每个字母代表一个色样），就可以进行相应的设置了（图 7-4）。可以被定制的色样内容包括：

- 页头 / 页脚栏的颜色，阴影以及背景；
- 正文内容的颜色和边框；
- 正常、悬停以及按下状态时，按钮的颜色和边框。

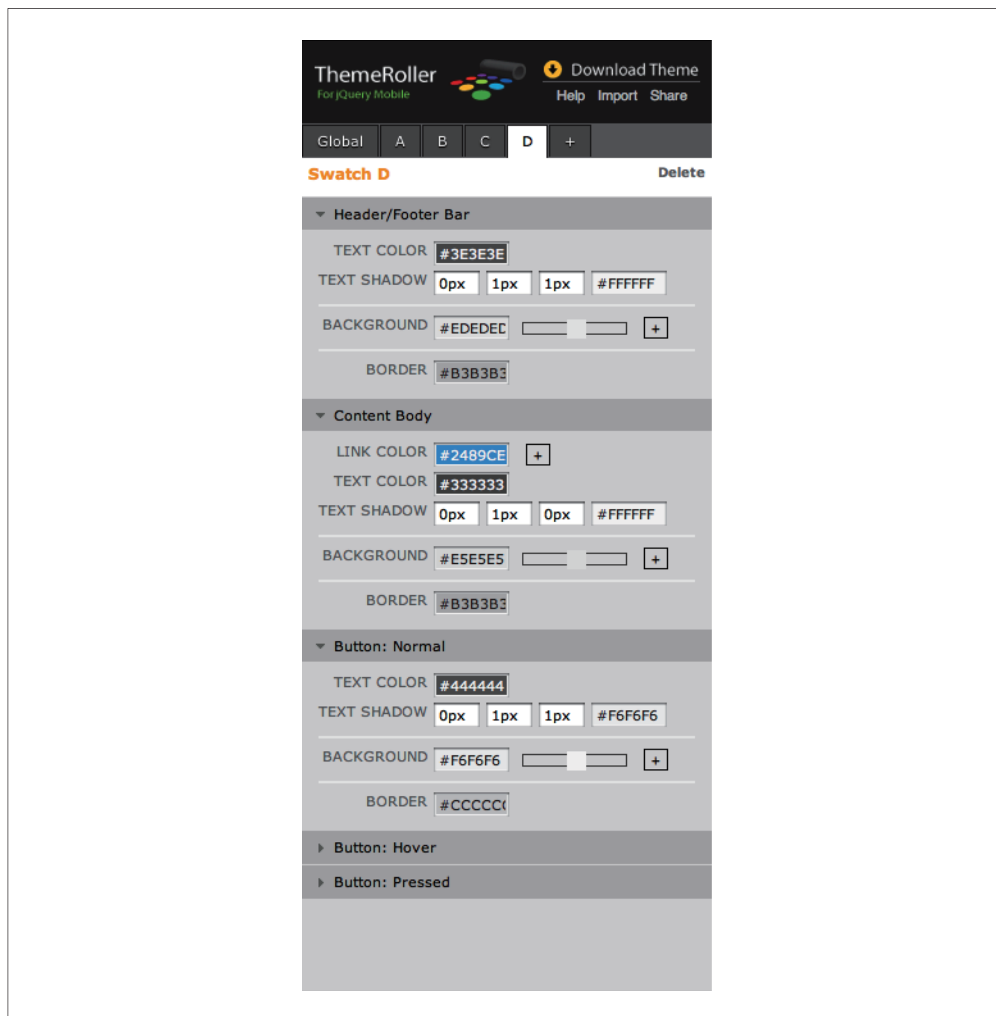


图 7-4：每个字母都代表了一整套对于工具栏 / 内容 / 按钮这些元素的颜色，效果和背景的定义

选择“+”标签页，可以为主题增加新的色样。



设置背景色时，你也许会注意到旁边有一个小滑块。移动滑块可以改变背景色的渐变，同时右边的预览窗格提供了实时预览。点击滑块右侧的加号可以进行更为详细的渐变设置。

7.1.3 审查器

把顶部面板上的审查器（Inspector）开关打开，然后在预览窗格中点选元素，就能看到选中元素的各项属性显示在左侧的色样窗格里。

7.1.4 Adobe Kuler

Adobe Kuler 是 Adobe 创建的一个分享配色方案的社区。人们在社区里分享自己最喜爱的配色，而这些配色方案可以通过 http://kuler.adobe.com 网站或者任一款 Creative Suite 应用程序来进行浏览。

jQuery Mobile 的 ThemeRoller 内置 Kuler 部件（图 7-5），允许用户浏览和搜索社区里的数千组配色方案，选中后拖放到预览面板上就可以直接使用了。



图 7-5：从 ThemeRoller 的顶部打开 Adobe Kuler，搜索社区里精彩的配色方案



如果想要从一张图片（比方说你的网站 logo）来获得配色方案，可以通过 http://kuler.adobe.com 上传你的图片生成配色方案，将它公开，这样你就可以从 ThemeRoller 里搜索到它了。

7.1.5 输出主题

完成主题的设计工作后，可以点击网页右上角的下载主题按钮来导出主题。只需要提供主题的名字（图 7-6），然后就能得到一小段代码，复制粘贴这段代码到应用中就可以套用这个主题。如果选择点击下载 Zip 压缩包，则会得到一个包含主题 CSS（已经被压缩过并且可以直接调试的）在内的压缩包文件。

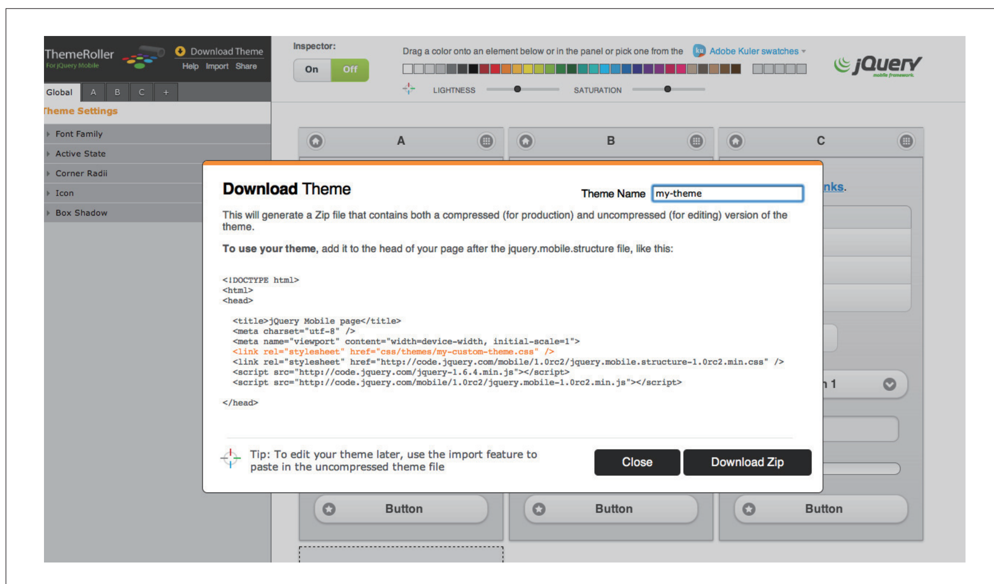


图 7-6：要导出主题，只需为它命名然后下载 Zip 压缩包文件即可



同样可以向 ThemeRoller 导入主题，这样就可以方便地通过 Web 界面对主题进行修改。选择 Import 选项，然后把 CSS 主题文件复制 / 粘贴进去即可。

7.2 Fireworks主题编辑器

Adobe Fireworks 用户可以很方便地使用 jQuery Mobile 主题编辑器。如果你的 Adobe Fireworks 是 CS5.1 版本，你可以从 http://labs.adobe.com 下载一个叫做“Fireworks CSS3 移动包”的免费插件[1]。

安装完插件包之后，打开 Fireworks 并且选择 Commands ➤ jQuery Mobile ➤ Create New Theme，就会见到如图 7-7 所示的一张图片。



如果没有 Adobe Fireworks，可以从 http://adobe.com/go/fireworks 下载一个 30 天试用版。

这张图片就像一个同名的可以修改的模板。打开页面面板 (Windows ➤ Pages) 可以看到图片被分割成了 6 个页面 (图 7-8)。

译注 1：Adobe 已不再提供这个包的下载，CS6 内置了 jQuery Mobile 主题编辑器。

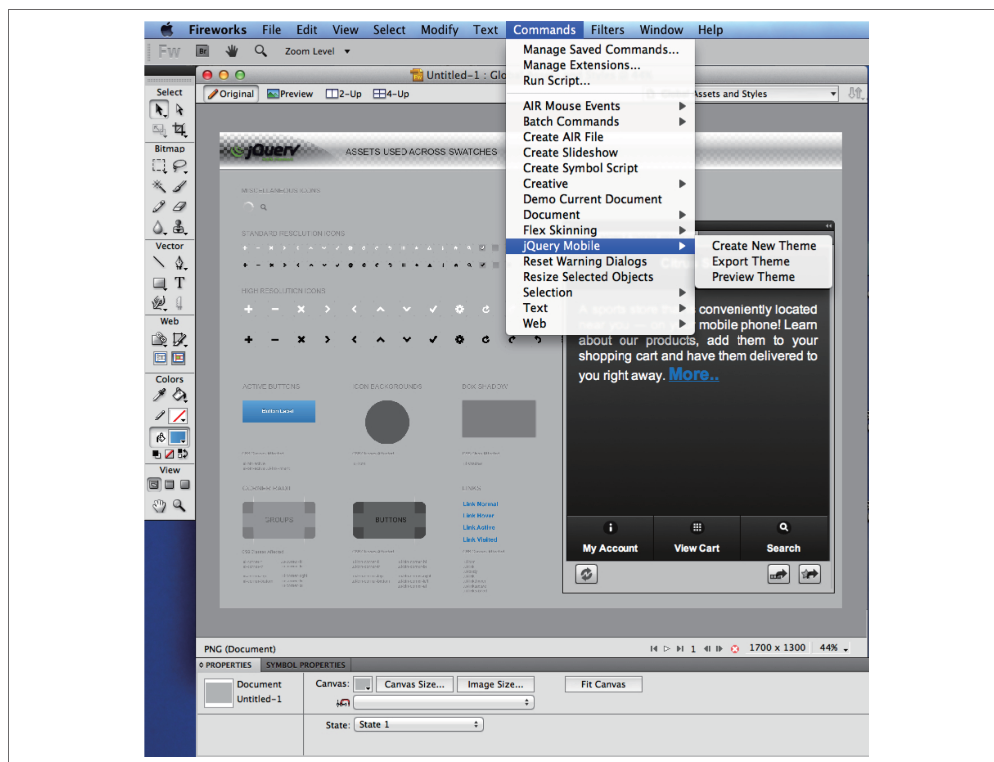


图 7-7：Adobe Fireworks 也可以用于创建 jQuery Mobile 主题

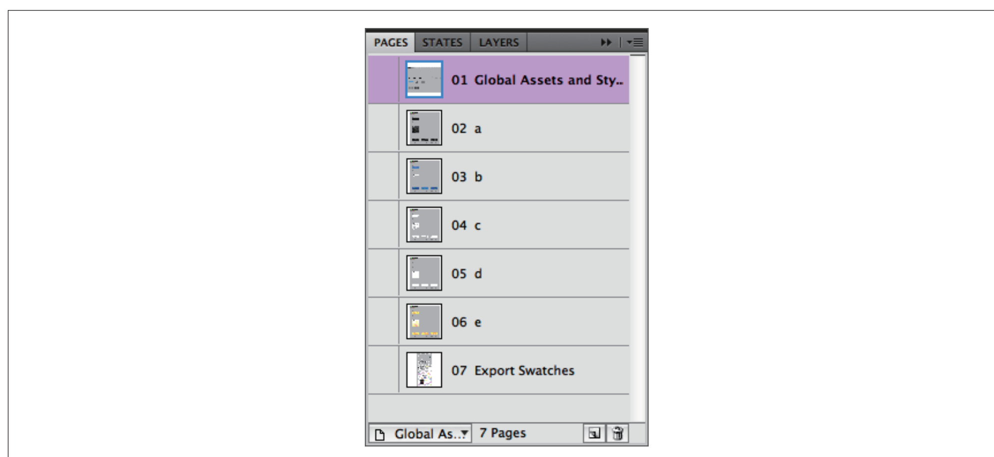


图 7-8：打开页面面板，可以看到一个全局设置页以及对应每个色样的设置页面

第一页叫做全局资产和样式，用于面向所有色样的全局样式和图标设置（图 7-9）。还有以字母 a 到 e 进行命名的页面，用于定义和改变各自对应的色样（图 7-10）。

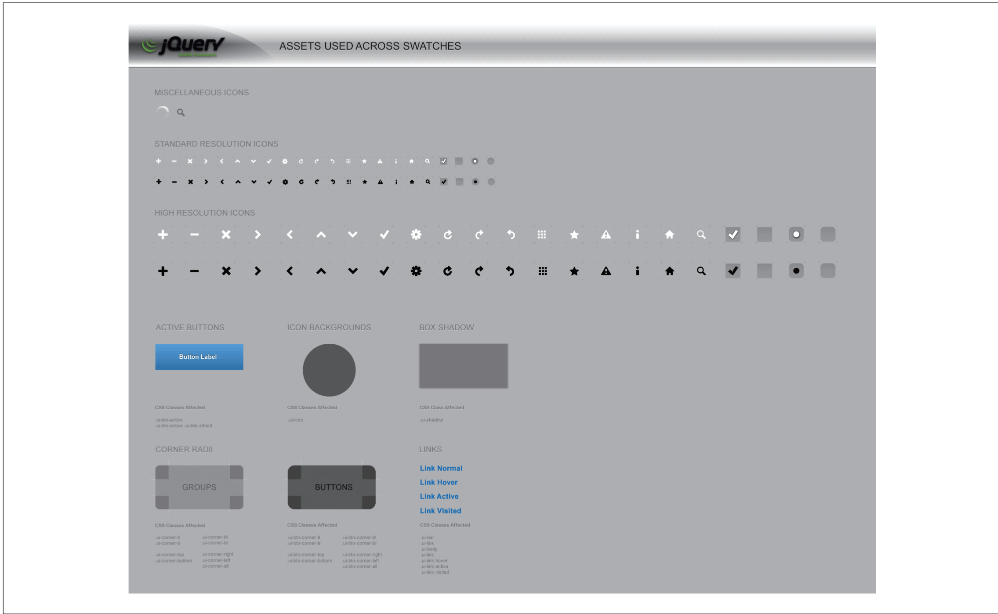


图 7-9：全局配置页允许用户改变图标，进行全局设置

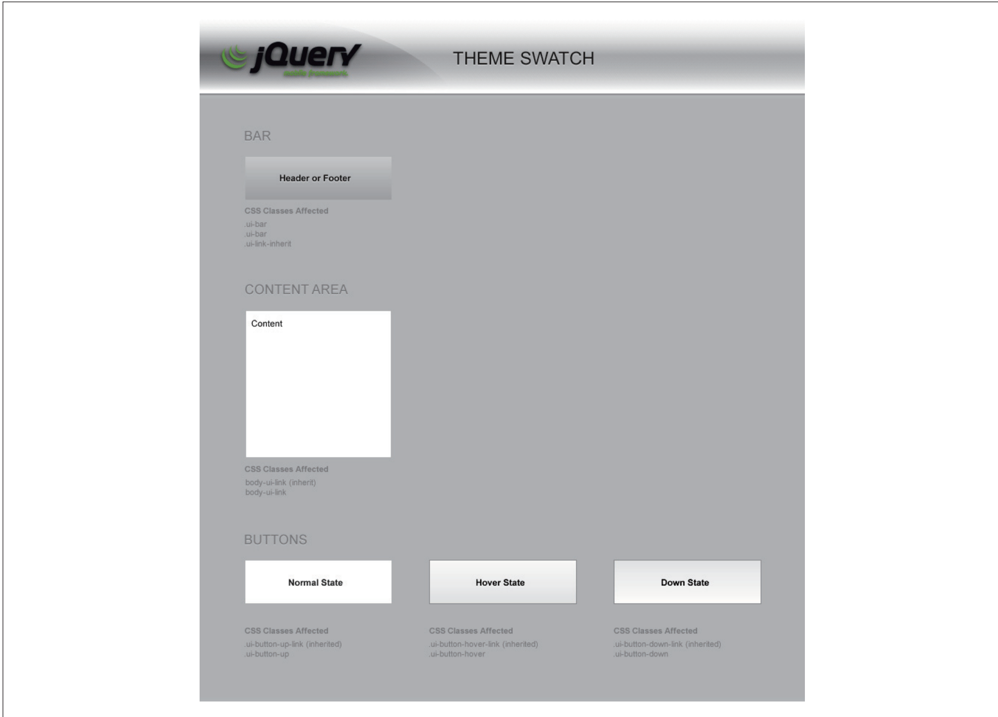


图 7-10：以字母 a 到 z 命名的页面，包含各自同名色样的定义

可以通过复制现有的设置页来增加更多的色样（鼠标右键点击并选择复制页面），然后选择一个合适的字母为它命名，比方说，f。不需要的色样可以删除，但建议至少保留 a 到 c 三个色样。

我们可以选择屏幕上的任意元素并用 Fireworks 来修改，可以被修改的元素属性包括：

- 文字颜色；
- 背景（纯色或者线性渐变色）；
- 字体样式和大小；
- 阴影过滤器；
- 不透明度 (透明度)。

在全局设置页面，我们可以定义的选项如下。

- 图标：低分辨率和高分辨率图标。你可以改变图标，但前提是保持名字不变，比如加号的名字是 ui-icon-plus。
- 激活按钮的样式。
- 图标背景：可以改变它的透明度。
- 盒阴影：可以改变阴影的颜色和形状。
- 控件组和按钮的圆角半径：每一个转角都可被选中，然后通过更改属性面板上的值来定义它的弧度。
- 各种状态下的链接：正常，悬停，激活，以及已访问状态。

在色样设置页面中可以定义：

- 页头和页脚区域；
- 内容区域；
- 各种状态下的按钮：正常，悬停以及按下。

完成主题修改后，可以选择 Commands ➤ jQuery Mobile ➤ Preview Theme 来预览主题。假使觉得还不错，可以选择 Commands ➤ jQuery Mobile ➤ Export Theme 来导出主题。

如图 7-11 所示，选择 Windows ➤ Extensions ➤ jQuery Mobile Theme Preview，可以打开 jQuery Mobile 预览面板。如果改变了当前页面的色样设置，你得刷新之后才能在预览面板中看到更新。

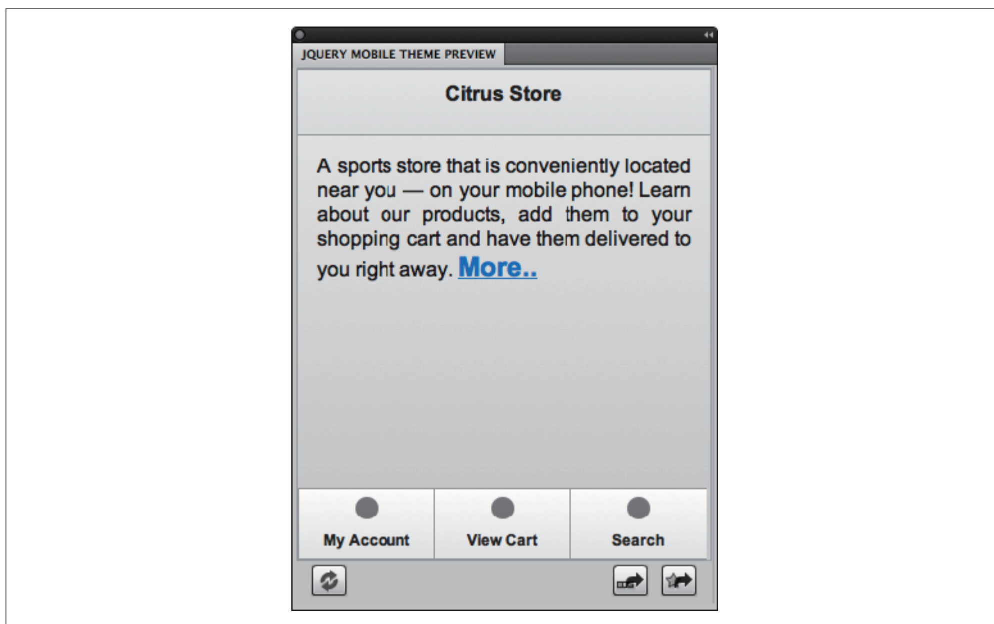


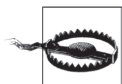
图 7-11：打开 jQuery Mobile Theme Preview，看看自定义的主题在真正的 Web 应用中长什么样儿



如果只是修改图标图片而不包括 CSS 文件，可以在 jQuery Mobile 主题预览面板找到一个输出图片精灵（Export Sprite）的图标。

Fireworks 会把设计输出成 CSS3 文件，包括设计中的各种渐变（已经自动为各个浏览器添加了相应的前缀），以及一张汇集所有图标的大图，框架会通过 CSS 图片精灵的方式从大图上定位图标。导出操作需要一个文件夹目录以及主题名，导出后可以得到一个 CSS 文件和一个图片目录，它们可以被包含到应用中直接使用。

推荐将这些文件保存成为一个 Fireworks PNG 文件，那么下次需要对主题进行微调时就可以直接使用这个 PNG 文件，而不是从头开始了。



如果打算使用自定义的主题 CSS，那就需要在 Web 应用中引入结构性 CSS 文件，而不是默认主题的 CSS。

7.3 编辑主题

主题就是一个 CSS 文件。这意味着我们可以在任何一款文本编辑器上从头创建或者修改主题。不过，要修改主题，就有必要了解 jQuery Mobile 定义元素和类的方式。

jQuery Mobile 使用类来定义样式。HTML 标记中的每一个小部件都会被转换成被若干 CSS 类所修饰的元素。因此，用户界面方面的工作就是为这些类定义样式。



注意，页面内容区域中的 HTML 代码不受主题的约束。在内容区域里使用不同于框架的 HTML 标记，用不同于框架 CSS 的样式来进行修饰都没有问题。

每个类名都有一个 `ui` 前缀，它的后缀是所套用的色样，类似于 `ui-< 名字 >-< 色样 >`。拿定义按钮来说，套用色样 `a` 的按钮的类名就是 `ui-btn-a`，套用色样 `c` 的按钮就是 `ui-btn-c`。然后，对于使用到 `data-theme` 或者其他相关属性的 HTML 标记来说，我们需要定义所要套用的色样。

表 7-1 展示了常见的可以在主题 CSS 文件中编辑和定义的类名。

表7-1：主题文件中可以修改从而定制用户界面的类

类　　名	描　　述
ui-bar-<x>	页头、页脚以及其他条栏
ui-btn-up-<x>	常态下的按钮
ui-btn-hover-<x>	悬停状态下的按钮
ui-btn-down-<x>	按下状态的按钮
ui-btn-active	激活状态的按钮（适用所有的色样）
ui-body-<x>	页面内容区
ui-link-<x>	链接
ui-icon-<x>	按钮和其他小部件用到的图标
ui-corner-all	应用于所有使用到圆角的控件
ui-corner-<tl/tr/bl/br>	应用于左上角 / 右上角 / 左下角 / 右下角的圆角
ui-corner-<top/bottom>	应用于顶部 / 底部的圆角
ui-corner-<left/right>	应用于左边 / 右边的圆角
ui-shadow	应用于所有需要显示阴影的元素
ui-disabled	应用于所有在 HTML 上被禁用的元素



如果想要为相同类的不同元素，比如页头和页脚（都使用了相同的类 `ui-bar`），提供不同的界面样式，应该在各自 HTML 标签里面改变各自所使用的色样。

7.4　定制页面过渡

要定制页面间的过渡，可以用前面章节讲过的 JavaScript 的方式，也可以完全使用 CSS3。如果打算使用 CSS3 动画，就有必要了解它的工作原理。

假如 data-transition 属性的定义无法被系统识别，它会转去寻找是否存在同名的 JavaScript 处理程序。如果还是找不到，它会尝试应用 CSS 动画。

就像是应用 CSS 类名一样，我们把过渡名同时应用到当前页和下一页。in 类会应用到下一页，而 out 类会应用到当前页。

也就是说，如果要定义一个名叫 card 的过渡，就需要为 .card.in 和 .card.out 各定义一个选择器。此外，你可以选择定义一个逆向过渡，用于页面返回。为此需要添加相应的 reverse 类，也就是还要定义 .card.in.reverse 和 .card.out.reverse。



我们并不需要提供动画的调速方法也不需要提供动画的持续时间，因为它们已经在全局的结构性 CSS 文件中定义了。

通过使用 CSS3 动画，可以创建个性化的页面过渡。卡片过渡的方式类似于幻灯片，但它的做法是先将所有的页面一层层地叠加起来，切换到下一张卡片时，只是移除了顶部的卡片（当前页面），然后下一页就显示出来了（没有动画）。

```
.card.out {
   -webkit-transform: translateY(-100%);
   -webkit-animation-name: cardout;
   z-index: 1; /* 在上面 */
}

.card.in {
   -webkit-transform: translateY(0);
   z-index: 0; /* 在下面 */
}

@-webkit-keyframes cardout {
   from {
       -webkit-transform: translateY(0%);
   }

   to {
       -webkit-transform: translateY(-100%);
   }
}
```

可以把这部分定义添加到主题 CSS 文件或者另一个单独的 CSS 文件中。要用的时候，只需通过 data-transition 属性或 JavaScript 定义它就可以了。

```
<a href="#page2" data-role="button" data-transition="card">Page 2</a>
```

要想让过渡动画也能在 Android 版 Firefox、Opera 或者 IE10 上运行，还需分别添加带各浏览器专用前缀（如 -moz、-o 以及 -ms）的样式。

第 8 章

安装以及离线访问

使用 HTML5 技术和其他扩展，可以让 jQuery Mobile 应用和移动设备上的原生应用一样完全离线运行。

此外，也可以把 jQuery Mobile 应用打包成一个原生应用通过应用商店发布，这在后面的章节会讲到。在这一章，我们将会学习创建一个不通过应用商店发布的离线 Web 应用。

使用这种方案，用户可以通过手机浏览器访问 Web 应用然后将它安装到手机上。那么下一次，当他通过同样的 URL 或者应用图标访问这个应用时，应用将不再从服务器端加载，而是直接从本地启动。

8.1 软件包定义

首先要做的事情是定义软件包。这里我们要用到 HTML5 的一个 API，叫做应用程序缓存（Application Cache），在 W3C 的草案中也被称为离线 API（Offline API）。

这个 API 并不是在所有的手机浏览器上都能用，但是它兼容大多数的平板电脑和智能手机。访问 http://mobilehtml5.org 可以检查 API 当前的兼容性。

在实际开始定义软件包前，需要明确目标。是打算构建一个完全离线的应用，还是希望每次只从服务器端获得针对某些页面或者某些数据的更新，又或者只是保存一份离线的数据缓存，而在有网络连接的情况下会进行更新？

明确目标之后就可以开始定义软件包了。软件包其实是一组文件，是用户访问应用站点时必须由浏览器下载到本地的文件。这组文件必须包含我们想要离线访问的每一个资源，比如 JavaScript、CSS 以及图片。我们正在创建 jQuery Mobile 应用程序，因此需要把所有的 jQuery Mobile 文件添加到软件包中，包括 CSS 结构文件、主题文件、图片和 JavaScript 文件。

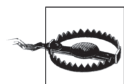
8.1.1 HTML清单

软件包内的文件是根据一个被称为缓存清单（cache manifest）的文本文件来发布的。清单文件的第一行必须是 `CACHE MANIFEST`，随后是所有需要被下载到设备上的资源的 URL 列表，URL 可以是相对路径或者绝对路径。

软件包内的 HTML 首页是默认的，并不需要在清单内声明。

清单中列举的文件是否存放在同一服务器上并不重要。也就是说，我们完全可以引用 jQuery Mobile CDN(内容分发网络) 上面的框架文件。

可以在行首使用 # 做行注释。



安装时，清单上的任一文件下载失败都会导致软件包无效。这意味着如果将资源部署在第三方服务器上，应用安装会依赖第三方服务器。

举例来说，一个典型的只包含一个 HTML 文档（没有外部页面）的 jQuery Mobile 应用清单看起来应该是这样的：

```
CACHE MANIFEST:

# jQuery 核心文件
http://code.jquery.com/jquery-1.6.1.min.js

# 没有自定义主题的 jQuery Mobile 文件
http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css
http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js
http://code.jquery.com/mobile/1.0/images/ajax-loader.png
http://code.jquery.com/mobile/1.0/images/icons-18-black.png
http://code.jquery.com/mobile/1.0/images/icons-18-white.png
http://code.jquery.com/mobile/1.0/images/icons-36-black.png
http://code.jquery.com/mobile/1.0/images/icons-36-white.png

# Web 应用自定义的文件，路径是相对于 HTML 文档的地址
images/logo.png
data/countries.json
```

清单文件的名字通常叫 offline.appcache，并且只有以 `text/cache-manifest` 的 `MIME` 类型提供给客户端才能生效。如果不知道如何设置 `MIME` 类型，你应该联系服

务器管理员。

如果服务器支持 PHP，那么只需要把文件的扩展名改成 .php，然后使用下面这个模版，不需要其他特殊配置就能生效了：

```
<?php header('Content-Type: text/cache-manifest');
?>CACHE MANIFEST:
```

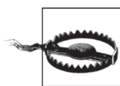
下一步需要在 HTML 文件中定义清单文件的 URL，这是通过设置 html 元素的 manifest 属性来完成的，这也是 HTML5 新引入的一个属性。

```
<html manifest="offline.appcache">
      <-- 应用的正文 -->
</html>
```

8.1.2 下载应用

支持应用缓存的浏览器找到清单声明后，就会在后台下载清单文件。如果清单文件有效并且 MIME 类型也有效，它就会启动 Web 应用的下载进程。

后台的下载进程和正常的页面加载是完全分开的。下载进程会把清单里面的每一个文件都下载下来，保存在设备上一个只有它自己可以访问的地方。



只要有一个资源没有下载成功（资源不存在或者服务器宕机），那么整个软件包都将无效，所有的资源都不会被保存在本地。

如果软件包安装成功，那么下一次用户访问相同 URL 时，浏览器就会加载应用的本地版本，而不再去访问服务器端。即使用户连上网络，浏览器也会自动地切换到离线状态。也就是说，在默认情况下，我们不能访问以前没有在清单文件中声明需要连线访问的资源。

如果软件包没有安装成功，页面下一次会重新从 Web 服务器加载。软件包没有安装成功的原因可能是：

- 清单文件无效，不存在，或者 MIME 类型错误；
- 至少有一个清单中列举的资源无法被访问到；
- 在所有的资源被正确地下载到本地之前，用户退出了浏览器或者网页。

使用调试工具可以查看软件包里包含的内容。针对 iOS 模拟器，可以下载免费的工具 iWebInspector，在“资源”选项卡里能找到应用程序的缓存信息（图 8-1）。

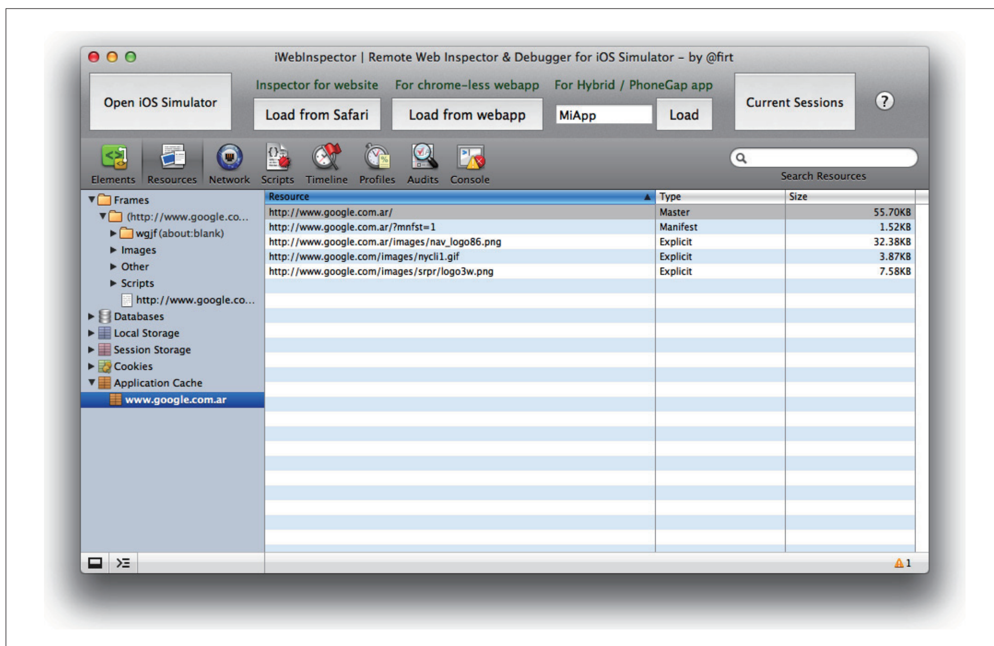
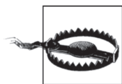


图 8-1：iWebInspector 的“资源”选项卡展示了应用缓存了的软件包内容



要访问应用下载的本地资源，并不需要修改代码。只需要像访问在线资源一样地访问它就可以了，即仍然使用与之前在清单中定义的相同的 URL 路径（相对地址或者绝对地址）。浏览器将智能地加载指定文件的本地版本。

8.1.3 访问在线资源

由于应用是运行在一个离线的沙箱内，所以无法访问先前没有在清单中定义的资源。如果清楚地知道需要从网上获取哪些信息，可以在清单文件中定义一个专门的 NETWORK：段落。默认情况下，所有的资源都定义在无需显式声明的 CACHE：段落内。段落是以冒号结尾的一行。如果希望每次都从服务器端获取 countries.json 文件，可以把清单改成：

```
CACHE MANIFEST:

# jQuery 核心文件
http://code.jquery.com/jquery-1.6.1.min.js

# 没有自定义主题的 jQuery Mobile 文件
http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css
http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js
http://code.jquery.com/mobile/1.0/images/ajax-loader.png
http://code.jquery.com/mobile/1.0/images/icons-18-black.png
```

```
http://code.jquery.com/mobile/1.0/images/icons-18-white.png
http://code.jquery.com/mobile/1.0/images/icons-36-black.png
http://code.jquery.com/mobile/1.0/images/icons-36-white.png

# Web 应用自定义的文件，路径是相对于 HTML 文档的地址
images/logo.png

# 需要每次都从网络上获取的资源
NETWORK:
data/countries.json
```

这样，countries.json 就不会被下载到本地，而是每次都从服务器端获取。没有连网时，这个文件就访问不到了，除非浏览器保存了一份缓存（指传统的浏览器缓存，而不是应用缓存）。



对于支持离线 API 的浏览器，有两个新增的可以操作文档对象的事件：online 和 offline。查询 window.onLine 这个布尔类型的属性，可以知道当前是否有互联网连接。

清单中的 NETWORK 段落支持使用通配符 * 或者文件夹的资源描述方式。如果使用文件夹，那么在离线模式下，指定文件夹下的所有资源都会从网上获取。

如果希望离线应用中的所有资源在有网络连接的时候都从网络上获取，可以这样定义：

```
NETWORK:
*
```

应用这个配置的话，就只有在 NETWORK：段落前声明的文件才会从离线软件包中加载，其他的所有资源都会从网络上获取。



清单文件也支持 FALLBACK：段落的定义，这块内容不在书中详细介绍了。简单来说，它用于为网络资源定义备用 URL，在没有互联网连接的情况下会重定向到备用 URL。

8.1.4 更新资源

之前说到在软件包安装之后，所有的资源（包括 HTML 首页）都只会从本地存储而不是服务器端进行加载 。很自然地，人们会问：如何更新资源呢？如果想要更新主题 CSS 文件，改一张图片，或者在 HTML 文档中加入一个新的页面链接，要怎么做呢？

好吧，我得承认之前说的“应用第二次启动时会完全从本地加载资源”这句话不完全对，或者说遗漏了一些什么。嗯，确实忽略了一些情况。当用户第二次（包括以后）打开 Web 应用时，应用总是从本地存储进行加载，但同时浏览器会在后台进程中尝试从服务器端获取最新的清单文件！

如果没有互联网连接，那便什么都不会发生，用户始终使用的是本地版本。如果网络连接可用，浏览器就会比较刚从服务器下载的新清单文件和之前随 Web 应用一起下载到本地的旧清单文件——它可是逐字节地进行比较。只要有一个字节改变了，之前的清单就失效了，它就会根据新的清单文件把每个资源重新下载一遍。

再把上面那段重新看一遍，看完了？好，我们再来详细解释一下。如果一个 CSS 文件的内容发生了改变，而文件名保持不变，那么清单文件也不会改变。因此，已经下载的文件不会进行任何更新。所以，在这种情况下，一定要修改清单文件，才能确保浏览器下载新文件。

因文件需要更新而修改清单的方式包括增加空格、修改资源的名字（加上版本信息），甚至可以是增加一行注释、包含一个随机数或者最后修改时间，例如：

```
CACHE MANIFEST
# webapp updated 2012-01-01
```

哪怕只有一个字节发生改变（比如日期），旧的清单就会失效，浏览器就会重新下载所有的文件。是的，所有文件。离线缓存 API 不支持单个文件更新！



配合 Web 存储 API 来使用应用缓存，可以建立一种只下载部分资源（比如 CSS、JavaScript，或者图片）到本地而无需重新加载整个 Web 应用的机制。

清单更新过程中还有另一个让人不快的地方：如果发现有更新，浏览器会重新下载清单中的所有资源。然而，这个下载是在后台进程中完成的，此时以前的文件还显示在屏幕上。因此，就算是有更新，如果用户不重新启动应用，他看到的仍然是老版本。下次启动的时候，安装包已经改变，才会加载新的资源。

这意味着如果清单发生了改变，加载页面两次才能用上新的版本。这个问题可以用事件机制来解决。

8.1.5 JavaScript对象

有一个全局 JavaScript 对象可以帮助我们了解应用缓存的状态。这个对象叫做 `applicationCache`，它的 `status` 属性值可以参考表 8-1。

表8-1：`applicationCache.status`属性的值

值	常　　量	描　　述
0	UNCACHED	页面第一次加载，或者没有可用的清单文件
1	IDLE	缓存闲置 1
2	CHECKING	本地清单文件正在比对服务器端的清单文件进行检查
3	DOWNLOADING	资源下载中（第一次或者升级）
4	UPDATEREADY	有更新已经被下载，但要到下一次加载时才生效

要兼容所有的浏览器，需要先检查 `applicationCache` 对象是否可用：

```
if (window.applicationCache!=undefined) {
      // 浏览器支持 applicationCache 的使用
}
```

可以通过常量比较来查询当前状态，例如：

```
if (window.applicationCache!=undefined) {
      // 浏览器支持 applicationCache 的使用
      if (applicationCache.status==applicationCache.UPDATEREADY) {
          // 有更新等待重加载后生效
      }
}
```

`applicationCache` 对象包含 `update()` 方法（强制发起更新检查）和 `swapCache()` 方法（从旧的资源缓存切换到新的资源，前提是新资源已经下载完毕）。不过，已经加载的 HTML 文档和资源只有调用 `history.reload()` 进行一次完全重载后才能真正生效。

8.1.6 事件

`applicationCache` 对象支持多种事件，通过处理这些事件可以帮助我们管理各种场景。例如，当用户第一次访问网站时，可以显示一个“正在下载”的消息直到资源下载完毕，这样用户就会等待，完成下载的几率就会增加。

表 8-2 列举了 `applicationCache` 对象上可以绑定的事件。

表8-2：可以绑定处理程序的`applicationCache`对象事件

事件名	触发时机
checking	浏览器正在检查清单文件
downloading	浏览器开始下载清单文件上的资源
progress	其中一个资源下载完毕（每个资源下载完后都会触发，因此可以据以创建下载进度条）

译注 1：所有的缓存文件已经保存入缓存。

（续）

事件名	触发时机
cached	第一次下载已经顺利完成
noupdate	和服务器端清单文件比较后发现没有可用的更新
updateready	存在可用的更新，并且新的资源文件已经下载完毕，重新加载后生效
error	下载资源过程重出现错误
obsolete	检查更新后，本地清单文件失效，web 应用从本次存储中删除，下一次不再进行离线访问

在常见的场景下，开发者需要：

- 捕获 downloading 事件，以便向用户显示下载消息，额外地还可以提供一个旋转着的下载动画；
- 捕获 progress 事件，制作一个进度条，见图 8-2；
- 捕获 cached 事件，隐藏下载信息并告知用户应用已经安装完毕；
- 捕获 error 事件，隐藏下载信息并告知用户错误原因；
- 捕获 updateready 事件，通知用户应用更新已经准备完毕，并征询用户是否要马上重新加载以访问新版的应用。

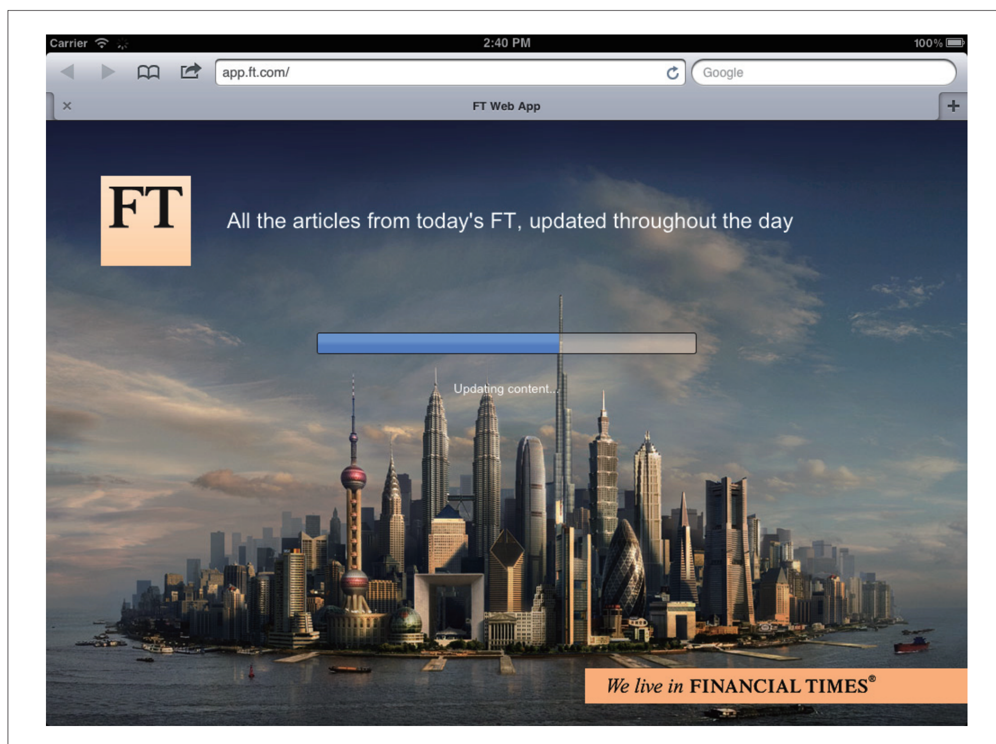


图 8-2：下载过程中的《金融时报》iPad 版应用程序（app.ft.com）

可以使用 addEventListener 方法来进行事件绑定，例如：

```
if (window.applicationCache!=undefined) {
    // 浏览器支持 applicationCache 的使用
    applicationCache.addEventListener('updateready', function() {
        // 有更新等待重加载后生效
        if (confirm("更新已经准备就绪，是否要立刻重新加载？")) {
            history.reload();
        }
    });
}
```

8.2 安装应用图标

应用下载完毕后，可以引导用户将应用图标添加到主屏幕或者应用菜单上。这样，用户就可以通过图标直接访问 Web 应用，无需重新键入应用的 URL 了。对于普通用户来说，在浏览器地址栏输入 URL 来打开一个离线应用（那种在坐飞机时还能使用的应用）是不可思议的。

即使没有清单文件（未启用应用缓存），用户也可以添加一个图标快捷方式到主屏幕上。在这种情况下，图标只是用户访问在线 Web 应用的一个快捷方式。



引导用户将 Web 应用快捷方式添加到浏览器书签中是另一种方案。不过，在移动应用的领域，大部分用户更普遍地选择安装桌面图标而不是加入浏览器书签。

目前还没有哪一个平台支持自动的应用图标安装，因此需要在屏幕上提供指引，引导用户自己安装。

8.2.1 引导

可以通过不同的方式引导用户将应用快捷方式添加到主屏幕上。YouTube、Google Maps 和 Facebook 在 iOS 上都通过如图 8-3 所示的浮动 div 来进行引导。

通常应用会将是否已经对用户进行过引导的记录保存在 cookie 或者 HTML5 localStorage 中，确保用户只会被引导一次。

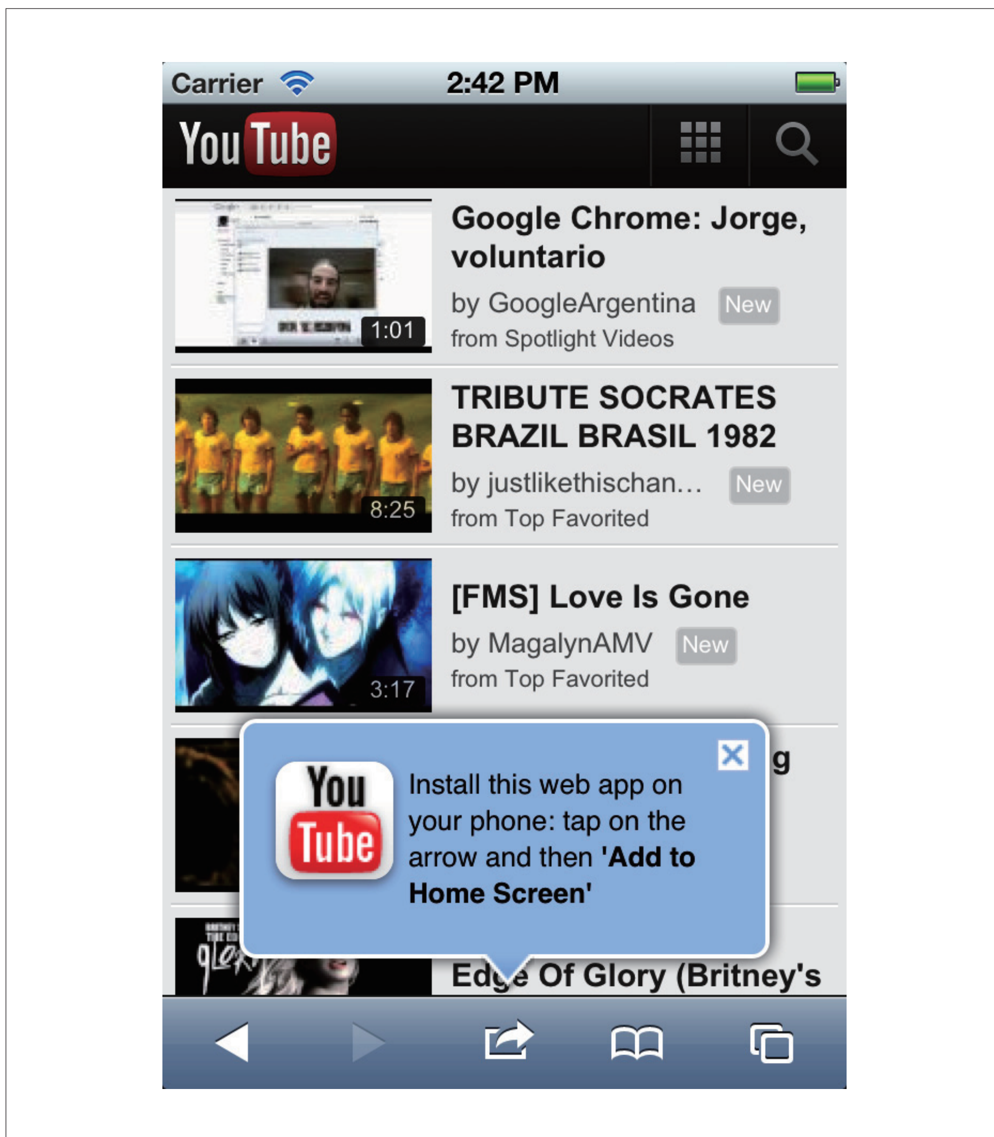


图 8-3：不少移动站点通过气泡提示的方式引导用户将应用程序快捷图标添加到主屏幕

8.2.2　图标快捷方式名

应用图标有一个对应的名字，在某些平台会显示在主屏幕上（图 8-4），在另一些平台则会显示在应用菜单上（图 8-5）。图标下方名字的默认值是 HTML 页面的 `title` 元素的值。如果打算引导用户安装应用快捷方式，那么有必要保持 `title` 的简洁（只用一两个词）以适应屏幕的大小。

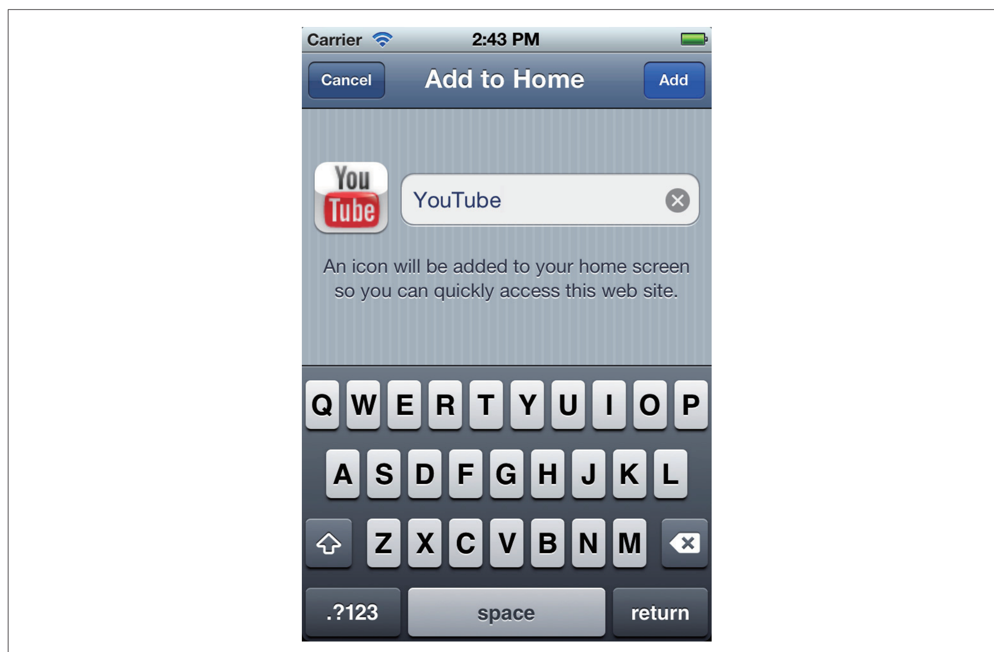


图 8-4：在 iOS 上用户可以将应用快捷方式添加到主屏幕上

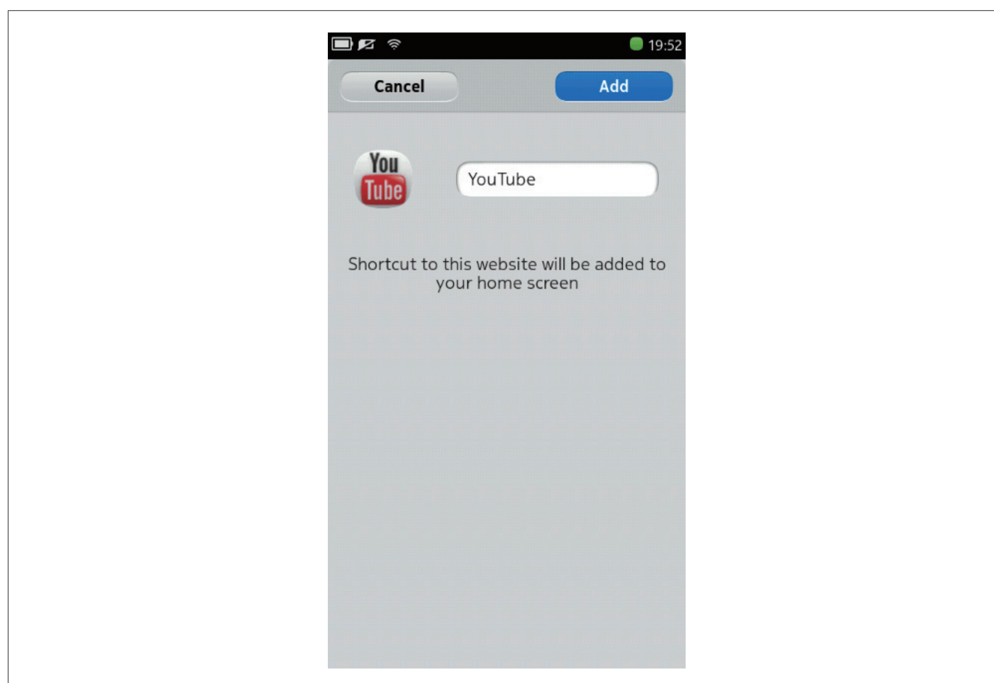
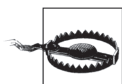


图 8-5：和 Android 手机以及 iPhone 一样，Nokia N9 也可以将应用快捷方式添加到主屏幕上



对于 jQuery Mobile 应用来说，如果用户在首页之外的页面（无论是应用内部还是外部的页面）上添加应用快捷方式到主屏幕或者应用菜单上，那么应用快捷方式会指向当前页而不是首页，当前页的 `data-title` 属性会被用作应用名。

图 8-6 展示了在不同平台上，如何将 Web 应用添加到主屏幕

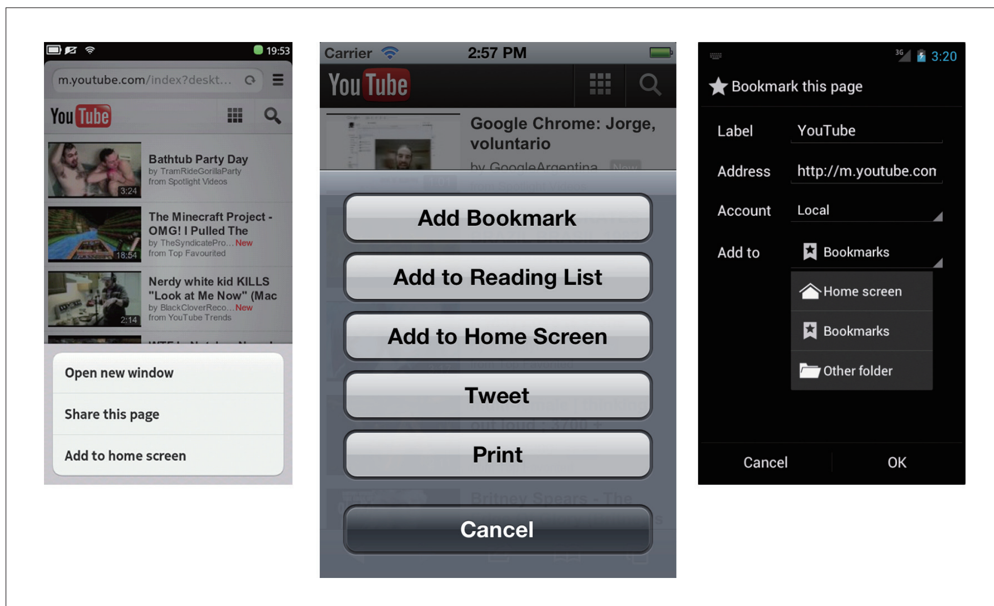


图 8-6：在各种平台上添加应用快捷方式到主屏幕或者应用菜单上的菜单

8.2.3　图标定义

传统网页通过 `favicon link` 元素指定 `rel="icon"` 或者 `rel="shortcut"` 来定义图标。所有的浏览器都支持这种定义方式。

```
<link rel="icon" type="image/png" href="favicon.png" />
<link rel="shortcut icon" type="image/png" href="favicon.png" />
```

这种方式定义的图标存在一个问题，即相比智能手机和平板电脑主屏上的图标，它们显得太小了。它们起初使用的是 ICO 格式（16 × 16 像素大小的位图图像），现在也支持其他格式，比如 PNG，但是通常都不大（最大也不过 32 × 32 像素）。这就是为什么这些图标通常不被用作应用程序快捷方式图标，即使用上了，也只是出现在实际图标的一个角落。



如果平台不支持主屏幕图标快捷方式，那么可以将 jQuery Mobile 应用打包成混合应用（比如使用 PhoneGap），然后作为原生应用在自己的网站或者应用商店上发布。这在后面几章会讲到。

除了上面说的，图标的定义没有其他的标准了。不过，苹果的 iOS 为它们自己的图标定义了一个 `meta` 标签，其他的平台现在也能使用它，比如 Android 和 MeeGo 1.2 上的浏览器（Nokia N9 的浏览器）。作为应用开发者，我们应该提供尽可能多的图标，以便各个平台使用各自支持的那个图标。如果平台没有找到它所支持的图标，那么通常它会使用标准的快捷方式图标或者网页左上角的截图——那个地方通常用来显示网站徽标。

作为事实上的标准，苹果定义的也是一个 `link` 标签，指定 `rel="apple-touch-icon"` 或者 `rel="apple-touch-icon-precomposed"`，而 `sizes` 属性是可选的。

默认情况下，苹果会为每一个指定 `rel="apple-touch-icon"` 的图标应用圆角、阴影和 3D 发光效果。如果不想启用阴影和 3D 发光效果（比如说一些透明的图标），那么可以设置 `rel="apple-touch-icon-precomposed"`。

Android 浏览器只支持 `rel="apple-touch-icon-precomposed"`，它不会为图标添加任何视觉效果。

可以为不同平台提供不同大小的图标。如果只提供一个图标，它会根据不同的平台缩放，效果可能不佳。比较好的做法是在网页内定义多个不同尺寸（使用 `sizes` 属性）的 `link` 元素，比如 `sizes="57x57"`。图标格式方面，PNG 是首选。

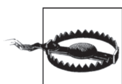
表 8-3 展示了各平台的图标尺寸。

表8-3：各平台主屏幕图标的尺寸

平台	尺寸	rel属性	sizes属性
iPhone3、3GS、iPod Touch 1-3	57×7	`apple-touch-icon` 和 `apple-touch-icon-precomposed`	从 iOS 4.2 开始支持
iPhone4、4S、iPod Touch 4	114×114	`apple-touch-icon` 和 `apple-touch-icon-precomposed`	从 iOS 4.2 开始支持
iPad 1、2	72×72	`apple-touch-icon` 和 `apple-touch-icon-precomposed`	从 iOS 4.2 开始支持
Android 浏览器	任何尺寸	`apple-touch-icon-precomposed`	不支持
Nokia N9 浏览器	80×80	`apple-touch-icon`	支持

Android 浏览器不支持 `sizes` 属性，已经定义的会被忽略。如果存在多个，Android 会使用最后一个定义 `rel="apple-touch-icon-precomposed"` 的那个图标。如

果希望 iOS 忽略供 Android 浏览器使用的图标，可以为链接元素提供一个无效的 `sizes` 属性，比如：`sizes="android-only"`。



低于 4.2 版本的 iOS 设备不支持 `sizes` 属性，如果定义了多个不同尺寸的图标，就会使用最后定义的那个。较好的做法是把尺寸较小的图标放在后面定义。

因此，可以预先存好所有需要用到尺寸的图标，然后在 `head` 元素内使用下面的这段代码来提供所有可能需要的图标：

```
<link rel="icon" href="icons/icon32.png">
<link rel="shortcut icon" href="icons/icon32.png">

<link rel="apple-touch-icon" href="icons/icon57.png" sizes="57x57">
<link rel="apple-touch-icon" href="icons/icon114.png" sizes="114x114">
<link rel="apple-touch-icon" href="icons/icon72.png" sizes="72x72">
<link rel="apple-touch-icon" sizes="80x80" href="icons/icon80.png">
<link rel="apple-touch-icon-precomposed" sizes="android-only" href=
    "icons/icon57.png">
```

图 8-7 展示了我们的应用在不同平台主屏上的图标。



图 8-7：为不同平台的主屏以及程序菜单应用不同尺寸的图标

8.3 全屏

在（且仅在）iOS 平台上（包括 iPhone 和 iPad），可以进一步定制我们的 Web 应用。我们可以创建一个没有边框的全屏应用，看不见任何的 Safari 控件（既没有地址栏也没有工具条）。不过，全屏显示要求用户必须从主屏幕上启动应用程序，并且在 HTML 页面上定义这样一个 `meta` 标签：

```
<meta name="apple-mobile-web-app-capable" content="yes">
```

然后，当用户打开页面时，就会看到一个完完全全的全屏应用（图 8-8），此时，有必要在界面上提供一个显式的后退按钮。

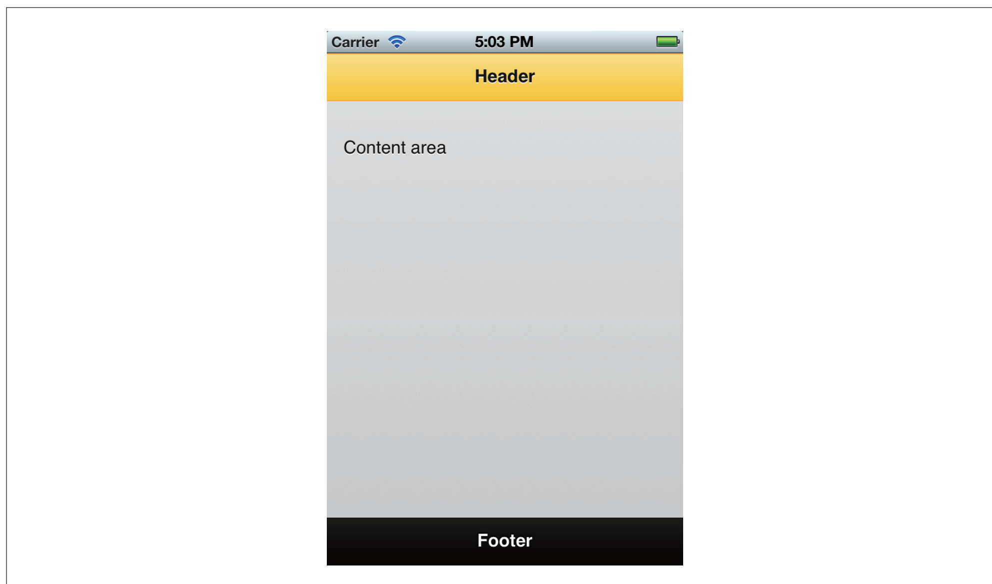


图 8-8：一个 jQuery Mobile 程序在 iPhone/iPad 上以全屏应用的方式打开

8.3.1 全屏检测

在 iOS 平台上，可以通过 `navigator.standalone` 属性强制程序全屏运行。如果这个属性可以被读到且值为 `false`，意味着用户是通过 Safari 访问应用程序，如果为 `true`，则应用是从主屏幕上的快捷图标启动的。

顺着这个思路，我们可以强制应用在全屏下进行安装和使用。一旦安装之后，用户无法从视觉上区分从 App Store 上下载的原生应用和我们的全屏应用。

比方说，可以用一个特殊的 jQuery Mobile 页面引导用户从主屏幕菜单上安装和使用我们的 Web 应用：

```
if (navigator.standalone!=undefined) {
    // 是 iOS 平台
    if (!navigator.standalone) {
       // 运行在 Safari 浏览器上
       $.mobile.changePage($("#install"), {transition: "none"});
    }
}
```

将上面这段代码放到 mobileinit 中，用户从 Safari 访问 Web 应用时，就会被重定向到安装页面而不是直接进入当前页面。如果用户是从主屏幕菜单访问，那么他就可以直接进入 jQuery Mobile 的初始化页面。

8.3.2 修饰Web应用

Apple 提供了一些 meta 标签用来修改 Web 应用的样式。首先是（屏幕的顶部）状态栏的颜色，可以通过定义 apple-mobile-web-app-status-bar-style 这个 meta 标签来修改，它的值可以是 default（灰色）、black 或 black-translucent。其中 black-translucent 是一个透明的黑色区域，它会显示一部分的应用标题颜色。图 8-9 展示了实际应用中的三种状态栏颜色。

```
<meta name="apple-mobile-web-app-status-bar-style" content="black">
```

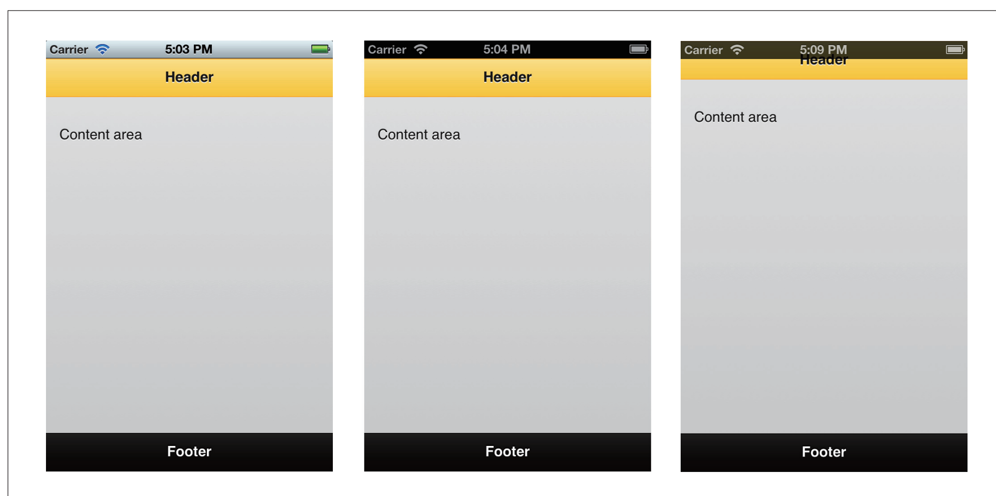
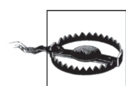


图 8-9：应用在同一页面上的不同状态条样式



如果使用 black-translucent 样式的状态栏，应用就可以占据浏览器的全部高度。也就是说，状态栏会悬浮在页面上方。如果不使用固定的工具栏并且没有在顶部留下 20 像素的空间，那么用户体验就会很糟糕。

全局应用另一个可以定制的地方是启动图片。启动图片是提供给操作系统用来在程序启动的动画中显示的图片，如图 8-10 所示。

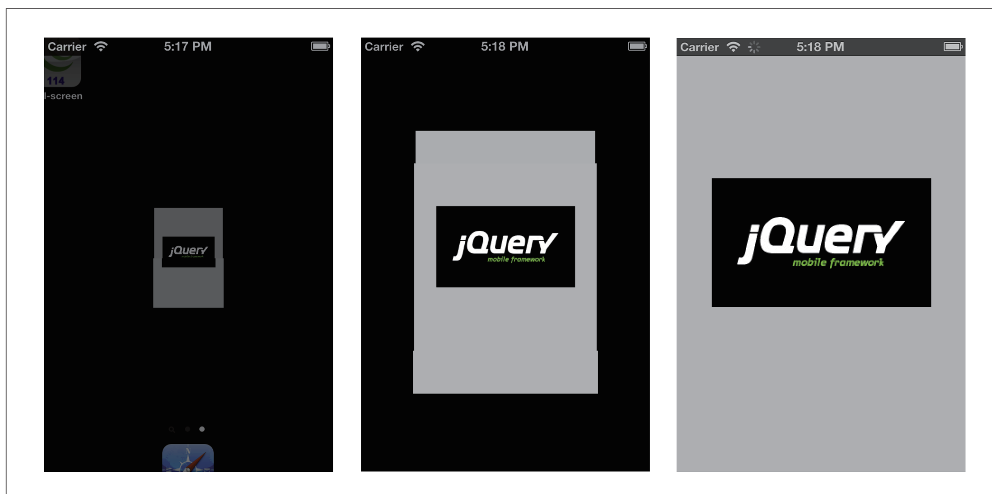


图 8-10：启动图片用在打开程序时所显示的动画中

启动图片可以通过指定 `rel="apple-touch-startup-image"` 的 `link` 元素来定义，它的尺寸依赖于运行的平台，可能是：

- 所有的 iPhone/iPod: 320 × 460 像素；
- 所有的 iPad 竖屏应用 : 768 × 1004 像素；
- 所有的 iPad 横屏应用 : 1024 × 748 像素。



遗憾的是，应用无法使用媒体查询来为 iPhone 4、4S 以及第四代 iPod Touch 增加一个高分辨率的启动图片。唯一可行的间接方案是使用 JavaScript 动态创建 `link` 元素，在这种情况下，所需要的图片尺寸是 640 × 920 像素。

如果只提供一幅图片，它的尺寸应该是 320 × 460 像素（20 像素留给顶部的状态栏），适用于所有的 iPhone 和 iPod Touch（包括高清版本）。但是它不适用于 iPad，系统会使用常规的启动图片（Web 应用最后一次使用的截图）：

```
<link rel="apple-touch-startup-image" href="images/launch.png">
```



如果你有一个 iOS 设备，可以同时按下主屏键和电源键来截图，并将其用作启动图片。这样，它就可以跟 Web 应用在第一次启动时的界面吻合了。

不同的启动图片尺寸可以定义在不同的 `link` 元素内，使用 `media` 属性的 CSS 媒体查询。和图标定义不同，它不会用到 `sizes` 属性。

```
<link rel="apple-touch-startup-image" href="images/launch-iphone.png"
    media="(max-device-width: 480px)">
<link rel="apple-touch-startup-image" href="images/launch-iPad-p.png"
    media="screen and (min-device-width: 481px) and (max-device-width:
    1024px) and (orientation:portrait)">
<link rel="apple-touch-startup-image" href="images/launch-iPad-l.png"
    media="screen and (min-device-width: 481px) and (max-device-width:
    1024px) and (orientation:landscape)">
```

8.4 完整的例子

一个提供离线访问，并且定义了快捷方式图标和全屏配置的 jQuery Mobile Web 应用的模版看上去应该是这样的：

```
<!DOCTYPE HTML>
<-- 包括应用程序离线缓存定义的 HTML -->
<html manifest="offline.appcache">
<head>
 <meta charset="UTF-8">
 <title>短标题</title>
 <meta name="viewport" content="width=device-width,user-scalable=no">

 <-- 使用自定义主题的 jQuery Mobile 文件 -->
 <link rel="stylesheet"
     href="http://code.jquery.com/mobile/1.0/jquery.mobile.structure-
          1.0.min.css" />
 <link rel="stylesheet" href="custom_theme.css">
 <script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
 <script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
 <script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js">
     </script>

 <-- 图标 -->
 <link rel="icon" href="icons/icon32.png">
 <link rel="shortcut icon" href="icons/icon32.png">

 <link rel="apple-touch-icon" href="icons/icon57.png" sizes="57x57">
 <link rel="apple-touch-icon" href="icons/icon114.png" sizes="114x114">
 <link rel="apple-touch-icon" href="icons/icon72.png" sizes="72x72">
 <link rel="apple-touch-icon" sizes="80x80" href="icons/icon80.png">
 <link rel="apple-touch-icon-precomposed" sizes="android-only" href=
     "icons/icon57.png">

 <-- 如果需要全屏展示 -->
 <meta name="apple-mobile-web-app-capable" content="yes">
 <meta name="apple-mobile-web-app-status-bar-style" content="black">
 <link rel="apple-touch-startup-image" href="images/launch-iphone.png"
     media="(max-device-width: 480px)">
 <link rel="apple-touch-startup-image" href="images/launch-iPad-p.png"
     media="screen and (min-device-width: 481px) and (max-device-width:
     1024px) and (orientation:portrait)">
 <link rel="apple-touch-startup-image" href="images/launch-iPad-l.png"
```

```
    media="screen and (min-device-width: 481px) and (max-device-width:
        1024px) and (orientation:landscape)">

</head>

<body>

<!-- jQuery Mobile 应用程序页面 -->

</body>
```

8.5 存储离线数据

HTML5 相关技术为离线应用（或者是那些不想使用 AJAX 的在线应用）的本地化信息存储提供了三种选择：

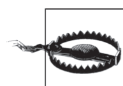
- Web Storage API；
- Web SQL Database API；
- IndexedDB API。



访问 http://mobilehtml5.org，可以检查每一个 HTML5 相关 API 的最新浏览器兼容性

本书选择 Web Storage API 来实现本地数据存储，一方面它最容易应用，另一方面它也是 jQuery Mobile A 级浏览器之中兼容性最好的。它支持 `localStorage` 和 `sessionStorage` 这两个基于键值对进行存储的集合。

`localStorage` 是一个在设备上持久存储的字符串集合，`sessionStorage` 也差不多，只是每次浏览器关闭后它会被清空。



通常情况下，每个主机地址可以使用至多 5Mb 的本地存储，这是没有任何问题的。然而，大多数浏览器都使用 Unicode 存储字符串，一个字符需要占用两个字节。所以，把存储的文本控制在 2.5Mb 以内是安全的。

`localStorage` 是全局 `window` 对象的属性，它有两个方法：`getItem` 和 `setItem`，分别对应从集合获取和向集合存储数据。

标准的 `localStorage` API 只能存储字符串，这意味着可以存储：

- 转换成 JSON 格式字符串的数组和对象；
- 简单格式的值；
- 逗号分割的值；

- JavaScript 代码（供随后解析执行）；
- CSS 样式表（供随后注入）；
- HTML；
- 转化成 base64 字符串的图片（data URI）。



主流的移动浏览器支持 JavaScript 对象和 JSON 字符串相互转化的 JSON API，使用 `JSON.stringfy(object)` 将对象转成 JSON 字符串，使用 `JSON.parse(string)` 将字符串转成 JavaScript 对象。如果浏览器不支持 JSON API，可以使用 Douglas Crockford 创建的 JSON2 库，它是免费的，在这里可以找到 http://github.com/douglascrockford/JSON-js。

如果要保存数据，可以这样写：

```
localStorage.setItem("name", "value");
```

如果要获取数据，可以这样写：

```
var value = localStorage.getItem("name");
```

第 9 章

Web应用实例

这一章，我们将会创建一个完整的 Web 应用。本书前面几章讲到的大部分内容，在这个实例中都会应用到。这是一个为某次大会开发的移动应用，它的主要目标是：

- 作为与会者的官方应用；
- 显示每个会议室的议程；
- 提供大会的各种信息。

如果你和我一样参加过很多技术会议，多半也有体会，会议现场的网络连接往往都不太可靠，所以我们的应用需要支持离线访问。我们将在后面章节介绍如何将其进一步转换成一个原生的离线应用。

9.1 Web应用的结构

这个应用的结构很简单，但是已经足够覆盖所有需要提供的功能，包括：

- 主页；
- 会议信息；
- 位置；
- 反馈；
- 关于。

会议信息列出所有有会议的时间段和每个会议室在各个时间段内会召开的会议。会议列表信息使用的是从服务器端发送到客户端的 JSON 对象，这些 JSON 对象是通

过 PHP 或者 Java 等服务器端语言生成的。关于服务器端如何生成 JSON 格式数据，以及如何通过内容管理系统来生成数据库方面的内容，我们在这里不展开介绍。

我们为 iPhone、iPod 和 iPad 用户提供的是一个全屏应用，因此会尝试劝导用户把应用安装到系统中。如果用户不安装，那么就为他们提供在线版本的 Web 应用。

应用包含如下文档：

- index.html（应用中用到的大部分页面都在这里定义）；
- feedback.html；
- feedback.php。

应用中用到的图片，放在了 `images` 目录里面：

- logo.png；
- sponsors.png；
- background.png；
- launch-ios.png（iOS 上的应用启动图片）；
- icon114.png（iPhone 高分辨率图标）；
- icon72.png（iPad 图标）；
- icon57.png（Android 以及 iPhone 低分辨率图标）。

当然，还需要放入所有 jQuery Mobile 框架自带的文件，包括它的 JavaScript、CSS 和图片文件。

应用中唯一需要定期更新的文件是 sessions.json，一个从服务器端获得的 JSON 格式的对象，由服务器端的脚本动态生成。基于性能和离线应用的考量，我们会使用 HTML5 的 Local Storage API 来缓存这个 JSON 文件。

9.1.1 离线清单

我们要创建的第一个文件是针对兼容设备的 HTML 缓存清单文件，Web 应用的离线文件都需要在其中声明。

应用的清单文件是这样的：

```
CACHE MANIFEST

CACHE:

# jQuery Mobile 框架文件
jquery.js
```

```
jquery.mobile.js
jquery.mobile.css
images/ajax-loader.png
images/icons-18-black.png
images/icons-18-white.png
images/icons-36-black.png
images/icons-36-white.png

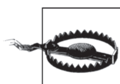
# HTML 文档
# index.html 始终会被缓存，无需显式的声明
feedback.html

# 自定义的脚本
index.js
index.css

# 自定义的图片
logo.png
sponsors.png
background.png

# 应用不需要缓存 iOS 图标以及启动图片，iOS 平台自己会缓存它们

NETWORK:
# 只有在请求 JSON 文件和获取反馈的脚本时，应用才会访问网络
# 把 mobilexweb.com 替换成自己的域名
http://mobilexweb.com/jqmbook/sessions.json
http://mobilexweb.com/jqmbook/feedback.php
```



我们没有在离线清单的 NETWORK: 段落中使用通配符 *。因此，一旦应用安装之后，就不会再访问互联网了。

9.1.2 页面

让我们从最重要的 index.html 开始。它是一个典型的 jQuery Mobile 文档，包含本书讲过的所有 HTML 头部标签（meta 标签、图标声明，以及视口声明），包含除 form 表单（它是最有可能每次都被用户访问的页面）之外所有的页面。我们可以在后期把一些内嵌的页面拆成外部的 HTML 文件。#sessions 页面是主列表页面，而 #details 页面是会议详细信息的模版文件。

在 #main 页面上，我们不打算使用页头，而是用会议的徽标代替之。其他页面上还是要使用页头。



我们不打算在 HTML 文档上提供会议的信息，而是通过 AJAX 从服务器端获取数据（封装成 JSON 对象）来把内容填到页面上。所以 #sessions 和 #details 源代码上看不到任何关于会议的信息。

下面是 index.html 文件的源代码：

```
<!DOCTYPE HTML>
<html manifest="manifest.appcache">
<head>
<meta charset="UTF-8">
<title>jQM 会议</title>
<meta name="viewport" content="width=device-width,user-scalable=no">
<link rel="stylesheet" href="jquery.mobile-1.0.css" />
<link rel="stylesheet" href="index.css" />
<script src="jquery.js"></script>
<script src="index.js"></script>
<script src="jquery.mobile-1.0.js"></script>
<meta name="apple-mobile-web-app-capable" content="yes">
<meta name="apple-mobile-web-app-status-bar-style" content="black">
<link rel="apple-touch-icon" sizes="80x80" href="icons/icon80.png">
<link rel="apple-touch-icon" href="images/icon57.png" sizes="57x57">
<link rel="apple-touch-icon" href="images/icon114.png" sizes="114x114">
<link rel="apple-touch-icon" href="images/icon72.png" sizes="72x72">
<link rel="apple-touch-icon-precomposed" sizes="android-only" href=
    "images/icon57.png">
</head>

<body>

<!-- **** 主页 **** -->

<div data-role="page" id="home">

  <div data-role="content">
    <p><img src="images/logo.png" alt="jQM Conference" width="300" height=
        "62" align="middle">
        </p>
    <h3>November 5th</h3>
    <div class="ui-grid-a">
      <div class="ui-block-a"><a href="http://www.twitter.com/fakeaccount"
          data-role="button" data-theme="b" target="_blank">Twitter</a>
          </div>
      <div class="ui-block-b"><a href="http://www.facebook.com/fakeaccount"
          data-role="button" data-theme="b" target="_blank">Facebook</a>
          </div>
    </div>
  </div>

  <div data-role="footer" data-position="fixed" data-id="toolbar">
    <div data-role="navbar">
      <ul>
        <li><a class="ui-btn-active" href="#home" data-icon="home"
                data-transition="fade">
                Home</a></li>
        <li><a href="#sessions" data-icon="grid" data-transition="fade">
                Sessions</a></li>
        <li><a href="#where" data-icon="info" data-transition="fade">Where
                </a></li>
```

```
        <li><a href="#about" data-icon="star" data-transition="fade">About
                </a></li>
        <li><a href="feedback.html" data-icon="plus" data-rel="dialog">
                Feedback</a></li>
      </ul>
    </div>
  </div>

</div>

<!-- **** 会议列表页面 **** -->

<div data-role="page" id="sessions">

  <div data-role="header">
    <h1>Sessions</h1>
    <a href='javascript:refresh();' data-icon="refresh" id="refresh"
        data-theme="c" class="ui-btn-left" data-iconpos="notext"></a>
  </div>

  <div data-role="content">
    <p id="console"></p>
    <ul data-role="list-view" data-inset="true" id="slots">
    </ul>
  </div>

  <div data-role="footer" data-position="fixed" data-id="toolbar">
    <div data-role="navbar">
      <ul>
        <li><a href="#home" data-icon="home" data-transition="fade">Home
                </a></li>
        <li><a class="ui-btn-active" href="#sessions" data-icon="grid"
                data-transition="fade">Sessions</a></li>
        <li><a href="#where" data-icon="info" data-transition="fade">
                Where</a></li>
        <li><a href="#about" data-icon="star" data-transition="fade">
                About</a></li>
        <li><a href="feedback.html" data-icon="plus" data-rel="dialog">
                Feedback</a></li>
      </ul>
    </div>
  </div>

</div>

<!-- **** 会议详细页 **** -->

<div data-role="page" id="details" data-add-back-btn="true">

  <div data-role="header">
    <h1>Sessions details</h1>
  </div>

  <div data-role="content">
    <div id="sessionInfo"> </div>
```

```
  </div>

  <div data-role="footer" data-position="fixed" data-id="toolbar">
   <div data-role="navbar">
    <ul>
     <li><a href="#home" data-icon="home" data-transition="fade">Home
             </a></li>
     <li><a href="#sessions" data-icon="grid" data-transition="fade">
             Sessions</a></li>
     <li><a href="#where" data-icon="info" data-transition="fade">Where
             </a></li>
     <li><a href="#about" data-icon="star" data-transition="fade">About
             </a></li>
     <li><a href="feedback.html" data-icon="plus" data-rel="dialog">
             Feedback</a></li>
    </ul>
   </div>
  </div>

</div>

<!-- **** 位置页面 **** -->

<div data-role="page" id="where">

  <div data-role="header">
    <h1>Where</h1>
  </div>

  <div data-role="content">
    Hasta la vista 1234, ACME 城

  </div>

  <div data-role="footer" data-position="fixed" data-id="toolbar">
    <div data-role="navbar">
      <ul>
        <li><a href="#home" data-icon="home" data-transition="fade">Home
               </a></li>
        <li><a href="#sessions" data-icon="grid" data-transition="fade">
               Sessions</a></li>
        <li><a class="ui-btn-active" href="#where" data-icon="info" data-
               transition="fade">Where</a></li>
        <li><a href="#about" data-icon="star" data-transition="fade">About
               </a></li>
        <li><a href="feedback.html" data-icon="plus" data-rel="dialog">
               Feedback</a></li>
      </ul>
    </div>
  </div>

</div>

<!-- **** 关于页 **** -->
```

```
<div data-role="page" id="about">

  <div data-role="header">
    <h1>About</h1>
  </div>

  <div data-role="content">
    <div data-role="collapsible">
        <h3>Orgnization</h3>
        <p>This congress is organized by ACME </p>
    </div>
    <div data-role="collapsible">
        <h3>Dates<</h3>
        <p>November 5th, 2015 9am to 6pm</</p>
    </div>
    <div data-role="collapsible">
        <h3>History<</h3>
        <p>First edition of this congress was.......</p>
    </div>
    <div data-role="collapsible">
        <h3>Sponsors</h3>
        <img src="images/sponsors.png" width="300" height="400">
      </div>
    </div>

    <div data-role="footer" data-position="fixed" data-id="toolbar">
      <div data-role="navbar">
        <ul>
          <li><a href="#home" data-icon="home" data-transition="fade">
              Home</a></li>
          <li><a href="#sessions" data-icon="grid" data-transition="fade">
             Sessions</a></li>
          <li><a href="#where" data-icon="info" data-transition="fade">
             Where</a></li>
          <li><a class="ui-btn-active" href="#about" data-icon="star"
             data-transition="fade">About</a></li>
          <li><a href="feedback.html" data-icon="help" data-transition=
             "fade">Feedback</a></li>
        </ul>
      </div>
    </div>

  </div>

  <!-- **** iOS 安装窗口页面 **** -->
  <div data-role="page" id="ios">

    <div data-role="header">
      <h1>installation</h1>
    </div>

    <div data-role="content">
      <p id="consoleInstall">installation</p>
      <div id="install">
```

```
            <p>To finish the installation, you must add this webapp to your
                Home Screen. To do that, touch the arrow and select "Add to
                Home Screen"</p>
      </div>
      <a href="javascript:openWithoutInstallation()" class=
          "openWithoutInstall">Open without installation</a>
    </div>

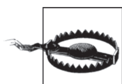
  </div>

  </body>
  </html>
```



我们使用了一个固定的页脚，以确保导航条在翻页时不会应用动画。

#sessions 页面上有一个 ID 为 console 的空的 p 元素，以及一个 ID 为 slots 的 listview 模块。我们会在页面加载后通过 JavaScript 向其中填入数据。#where 页面中的 map 元素的用途也是一样的。



如果要使用 navbar，我们需要在每个页面（无论是内嵌还是外部的）都把它包含进来，放在页头或者页脚里。如果不想要重复的代码，可以用 JavaScript 来为每个页面插入 navbar。各页面 navbar 上唯一可以改变的是里面被选中的元素。

提交反馈的表单是一个对话框，它的代码在 feedback.html 中，源文件如下：

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8" />
<title>jQM 会议 </title>
<meta name="viewport" content="width=device-width,user-scalable=no" />
<link rel="stylesheet" href="jquery.mobile-1.0.css" />
<script src="jquery.js"></script>
<script src="script.js"></script>
<script src="jquery.mobile-1.0.js"></script>
</head>

<body>

<div data-role="dialog">

  <div data-role="header">
    <h1>Feedback</h1>
  </div>

  <div data-role="content">
```

```
    <form action="feedback.php" method="post" data-transition="none">

      <div data-role="fieldcontain">
        <label for="name">Name:</label>
        <input type="text" name="name" id="name" value="" required />
       </div>

       <div data-role="fieldcontain">
         <label for="email">Email:</label>
         <input type="email" name="email" id="email" value="" required />
       </div>

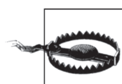
       <div data-role="fieldcontain">
         <label for="comments">Comments:</label>
         <textarea cols="40" rows="8" name="comments" id="comments">
             </textarea>
       </div>

           <div data-role="fieldcontain">
         <label for="contacted">Can we contact you?</label>
         <select name="contacted" id="contacted" data-role="slider">
           <option value="no">No</option>
           <option value="yes">Yes</option>
         </select>
       </div>

      <input type="submit" value="Send" data-theme="a" />

    </form>
  </div>

</div>
</body>
</html>
```



这里不详细讲述客户端以及服务器端的数据验证。但是记住，在对数据进行处理前一定要先验证。

feedback.php 会对数据进行验证，然后把数据存入数据库或者通过邮件发送出去，最终展示一个结果页面：

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8" />
<title>jQM Conference</title>
<meta name="viewport" content="width=device-width,user-scalable=no" />
<link rel="stylesheet" href="jquery.mobile-1.0.css" />
<script src="jquery.js"></script>
<script src="script.js"></script>
<script src="jquery.mobile-1.0.js"></script>
</head>
```

```
<body>

<div data-role="dialog">

  <div data-role="header">
    <h1>Feedback</h1>
  </div>

  <div data-role="content">

    <?php
        // Validation and processing here

        ?>
    Thanks for your feedback.

    <a data-role="button" data-inverse="true" href="index.html">Close</a>

  </div>

</div>
</body>
</html>
```

9.1.3 样式

index.css 中包含了自定义的样式：

```
/* 应用于页头的样式 */
.ui-header {
        background-color: #69C;
        background-image: url('images/background.png');
        background-position: top right;
        background-repeat: no-repeat
}

/* 离线应用安装过程中使用的样式 */
#console, #consoleInstall {
        background-color: #FF6;
        border-radius: 10px;
        -webkit-border-radius: 10px;
        padding: 5px;
        margin: 0;
        text-align: center;
        border: 1px solid #CCC;
        font-size: small;
}

/* 主页面内容区域 */
#home [data-role=content] {
        background-color: white;
        text-align: center;
```

```
}

/* 按钮 */
.openWithoutInstall {
        margin-top: 50px;
        display: block;
}

/*  导航条文字样式 */
[data-role=navbar] .ui-btn-text {
      font-size: smaller;
}
```

9.1.4 数据

包含会议信息的 JSON 文件的格式可以参考下面这个例子：

```
{
  "slots":
   [
     // 有的时间段是特殊时间，如开放时间、休息时间、午餐时间，等等
     { "id": 1, "time": "09:00", "message": "Opening" },
     // 有的时间段是会议时间
     { "id": 2, "time": "09:20"}
   ],
  "sessions":
   [
        { "id": 1, "title": "A great session", "timeId": 2,
          "description": "…", "speaker": "…", "room": "…" }
   ]
}
```

9.1.5 脚本

有一些功能必须要在 JavaScript 中来实现，比如：

- iOS 全屏探测；
- 会议信息下载；
- 会议列表以及会议详细信息页面的动态生成。

index.js 中的代码如下：

```
var data;

$(document).bind("mobileinit", function () {

        if (navigator.platform == "iPhone" || navigator.platform == "iPad"
                || navigator.platform == "iPod" || navigator.platform ==
                "iPad" || navigator.platform == "iPhone Simulator") {
```

```
                // 这是一个 iOS 设备，检测现在是否在全屏模式下
                if (!navigator.standalone) {
                        showIOSInvitation();
                }
        }

        /* 捕捉页面加载行为，动态确认会议信息 */
        $("#sessions").live("pageshow", function () {

                if (window.localStorage != undefined) {
                        // 有可用的 HTML5 本地存储
                        if (window.localStorage.getItem("data") !=
                                        undefined && window.localStorage.
                                        getItem("data") != null) {
                                // 在检测更新的同时，首先加载离线存储的会议信息
                                showSessions(window.localStorage.getItem
                                    ("data"));
                                $("#console").html("Checking data update");
                        } else {
                                // 没有本地存储数据
                                $("#console").html("Downloading session
                                    data...");
                        }
                } else {
                        // 不支持 HTML5 本地存储，每次都从服务器端下载 JSON 数据
                        $("#console").html("Downloading session data...");
                }

                loadSessionsAjax();
        });

});

function showIOSInvitation() {
 setTimeout(function () {
        // 隐藏保存信息，直到 Web 应用下载完成
        $("#install").hide();
        // 打开 iOS 提示，引导用户创建首页快捷图标
        $.mobile.changePage($("#ios"), {transition: "slideup",
            changeHash: false});
 }, 100);

 // 捕捉 HTML5 应用缓存事件，提供有用的信息
 if (window.applicationCache != undefined) {
        window.applicationCache.addEventListener('checking', function () {
                $("#consoleInstall").html("Checking version");
        });

        window.applicationCache.addEventListener('downloading', function () {
                $("#consoleInstall").html("Downloading application.
                    Please wait...");
```

```
        });

        window.applicationCache.addEventListener('cached', function () {
                $("#consoleInstall").html("Application downloaded");
                $("#install").show();
        });

        window.applicationCache.addEventListener('updateready', function () {
                $("#consoleInstall").html("Application downloaded");
                $("#install").show();
        });
        window.applicationCache.addEventListener('noupdate', function () {
                $("#consoleInstall").html("Application downloaded");
                $("#install").show();
        });
        window.applicationCache.addEventListener('error', function (e) {
                $("#consoleInstall").html("There was an error downloading
                    this app");
        });
        window.applicationCache.addEventListener('obsolete', function (e) {
                $("#consoleInstall").html("There was an error downloading
                    this app");
        });
        }

}

function loadSessionsAjax() {
 // 把 JSON 对象用文本表示，更容易做本地存储
   $.ajax( /*"http://www.mobilexweb.com/congress/*/ "sessions.json", {
     complete: function (xhr) {
         if (window.localStorage != undefined) {
                 if (window.localStorage.getItem("data") != undefined &&
                     window.localStorage.getItem("data") != null) {
                         if (xhr.responseText == window.localStorage.
                                 getItem("data")) {
                                 // 没有新的会议信息
                                 $("#console").html("Schedule is updated");
                         } else {
                                 // 有新的会议信息
                                 if (confirm("There is an update in the
                                     schedule available, do you want to
                                     load it now?")) {
                                         $("#console").html("Schedule is
                                             updated");
                                         showSessions(xhr.responseText);
                                 } else {
                                         $("#console").html("Schedule will
                                             be updated later");
                                 }
                         }
```

```
                } else {
                        // 支持本地存储，但是没有之前保存的缓存
                        $("#console").html("Schedule is updated");
                        showSessions(xhr.responseText);
                }
        } else {
                // 不支持本地存储，显示的信息不会进行保存
                $("#console").html("Schedule is updated");
                showSessions(xhr.responseText);
        }
  },
  dataType: 'text',
  error: function () {
        $("#console").html("Schedule update could not be downloaded");
  }
 });
}

var isFirstLoad = true;

function showSessions(string) {
        if (window.JSON != undefined) {
                data = JSON.parse(string);
        } else {
            data = eval("(" + string + ")");
        }
        if (window.localStorage != undefined) {
                // 把对象保存成字符串格式
                window.localStorage.setItem("data", string);
        }

        $("#slots").html("");

        var html = "";
        for (var i = 0; i < data.slots.length; i++) {

                if (data.slots[i].message != null) {
                      // 这是一个特殊的时间段，因此创建一个分隔
                      html += "<li data-role='list-divider' data-
                          groupingtheme='e'>" + data.slots[i].time + ": "
                          + data.slots[i].message + "</li>";
                } else {
                      // 正常的有会议的时间段
                      html += "<li><a href='javascript:showDetails(" +
                          i + ")'
                          >Sessions of " + data.slots[i].time + "</a></li>";
                }
        }

        $("#slots").html(html);
```

```
        if (isFirstLoad) {
                $("#slots").listview();
                isFirstLoad = false;
        } else {
                $("#slots").listview('refresh');
        }

}

function showDetails(index) {
        $("#details h1").html("Sessions of " + data.slots[index].time);
        var html = ""
        for (var i = 0; i < data.sessions.length; i++) {
                if (data.sessions[i].timeId == data.slots[index].id) {
                        // 为每个会议创建一个可折叠的组件
                        html += "<div data-role='collapsible'>";
                        html += "<h3>" + data.sessions[i].title + "</h3>";
                        html += "<h3>" + data.sessions[i].room + "</h3>";
                        html += "<h4>Speaker/s: " + data.sessions[i].speaker;
                        html += "</h4>";
                        html += "<p>" + data.sessions[i].description + "</p>";
                        html += "</div>";
                }
        }
        // 提供前往详细页的信息
        $("#sessionInfo").html(html);
        $("#sessionInfo div").collapsible();

        // 导向详细页面
        $.mobile.changePage($("#details"));
}

function refresh() {
        $("#console").html("Verifying…");
        loadSessionsAjax();
}

function openWithoutInstallation() {
        // 移除类似于对话框的 iOS 安装页面
        $.mobile.changePage($("#home"), {transition: "slideup",
            reverse: true });
}
```



针对 Android 或者其他平台，我们可以创建混合应用或者页面部件来提供相似的安装页面，链向应用服务器或者应用商店。

这个应用还有很多可以改善的地方。比如，可以为每个会议创建单独的像 `#details!22` 这样的 URL。那么，我们就可以根据这个地址，为内部链接提供动

态的详细内容加载。图 9-1 展示了在 iOS 系统上我们这个基于浏览器的 Web 应用的效果就像全屏程序一样。

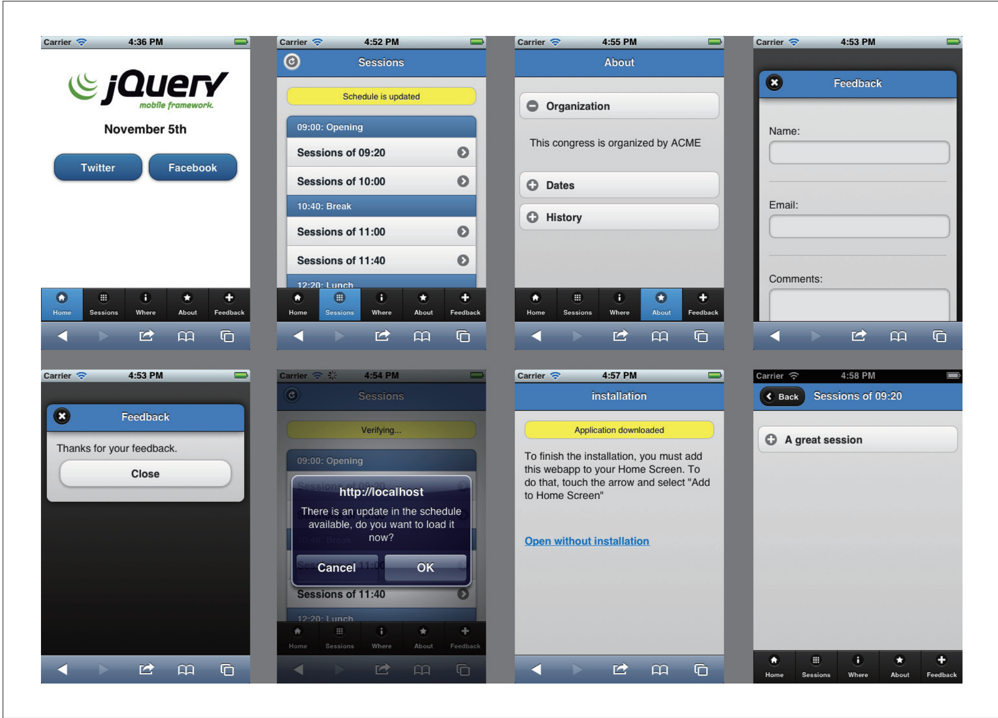


图 9-1：应用截图

第 10 章

扩展框架

jQuery 因其活跃的插件社区而被业界广为称道。同样地，jQuery Mobile 也可以很方便地经由第三方插件进行扩展。你可以在 http://jquerymobile.com/resources/#Plugins 找到最新的 jQuery Mobile 插件。

jQuery Mobile 支持多种形式的扩展。

- *主题*
  一个 CSS 文件，或一组图片，或者两者兼有。
- *插件*
  一个 JavaScript 文件，一个 CSS 文件，为框架增添新的部件 (`data-role`)。
- *扩展*
  一个 JavaScript 文件，为现有的 jQuery Mobile 部件和核心功能增加新的行为。

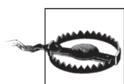
10.1 创建插件

jQuery Mobile 是基于 jQuery UI 的框架进行构建的，因此，为 jQuery Mobile 创建插件的方式和为 jQuery UI 创建插件差不多。

要创建具备良好兼容性的插件，首先需要认识到以下问题。

- 应该延续 jQuery Mobile 具备良好兼容性的传统，那些只兼容少数平台的插件对社区价值有限。

- 用户在使用插件时，只需在页面插入一个 JavaScript 文件和一个 CSS 文件。
- 尽可能地使用 data-* 属性，以延续框架的风格。
- 插件应能无缝地兼容现有的自定义命名空间以及主题切换。
- 插件应和框架中的其他部件一样自动进行初始化。
- 尽可能地使用带有语义的 HTML5 标签定义插件内容。
- 如果可能的话，支持无障碍访问（比如 ARIA）。
- 尽可能地避免使用通用的名字来命名自定义的 data-role，以避免与框架后续版本中的命名发生冲突。例如，不要用 tableview 或者 datepicker 这样的命名，因为后续版本的框架可能会用到这些名字。



HTML5、CSS3、ARIA 等内容已经超越了本书的范畴。但是要创建一个具备良好兼容性的优秀插件，必须要能够很好地掌握这些技术。

10.1.1　基础模板

部件可以通过以下两种方式来初始化。

- 自动初始化，即借助 data-role 或者带有语义的 HTML5 代码的声明，自动地初始化部件。
- 显式地通过 jQuery 语法调用部件的构造函数进行初始化，例如：$("#myelement").widgetname()。

创建部件要做的第一件事是命名，应尽可能避免和未来的框架部件重名。

部件应该包含一个命名格式为 jquery.mobile.<plugin-name>.js 的 JavaScript 文件，还可以包含一个格式为 jquery.mobile.<plugin-name>.css 的 CSS 文件。

部件的模板是这样的：

```
(function($){
     // 把所有的代码封装在这个方法中以确保 $ 是 jQuery 对象的引用

     // 部件的定义
     $.widget("mobile.ourWidgetName", $.mobile.widget, {
            options: {
            // 创建部件默认的选项
            },
            // 私有方法
            _create: function() {
                // 构造函数
            },


            // 公有方法
```

```
            enable: function() {
                // 启用部件
            },
            disable: function() {
                // 禁用部件
            },
            refresh: function() {
                // 刷新部件
            }
        });
        // 部件定义结束

        // 自动初始化代码
        $(document).bind("pagecreate", function(event) {
            // 找到相应的data-role 然后应用初始化方法
            $(event.target).find(":jqmData(role='ourWidgetName')").ourWidgetName();

        });

    }(jQuery));
```

10.1.2 创建插件

让我们通过创建一个部件来实际地展示一下创建插件的过程。这个部件的功能是动态提供不同尺寸的图片。它基于 Sencha IO Src 提供的云服务（免费的），这个服务能够根据当前移动浏览器的大小帮你调整图片的尺寸。



从 http://mobilexweb.com/go/senchaio 可以找到 Sencha IO Src 服务的官方文档。

我们把这个部件叫做 `dynamicimage`。它有两种实现方式，第一种是找到页面上的所有 `img` 元素自动地应用 `dynamicimage` 调整大小；第二种是定义一个新的 `data-role`，叫做 `dynamicimage`，只针对 `data-role` 为 `dynamicimage` 的 `img` 元素来调整大小。在这个例子中我们选择第二种方式。

1. 用法

整个思路是这样的，我们需要定义一个包含 `data-src` 属性的 `img` 元素，这个元素随后会加载 Sencha IO Src 提供的调整过大小的图片。不使用标准的 `src` 属性是为了避免重复加载原图和调整过大小的图片。

此外还需要定义两个特殊的属性，`data-width` 用来定义图片占据设备屏幕的百分比，`data-margin` 用来定义图片离屏幕窗口的边距。

由于 jQuery Mobile 没有为图片提供视觉特效，所以我们没有在图片上应用主题。

2. 部件

每个部件都是一个带有属性和方法的 JavaScript 对象。以下划线为首字母命名的方法是私有方法。在方法内部，`this` 指代的是当前部件对象，而 `this.element` 指代的是部件所对应的 HTML 元素。

```
$.widget("mobile.dynamicimage", $.mobile.widget, {
    options: {
        width: "100%",
        margin: 0
    }
});
```

`_create` 方法是部件的构造器。它只在初始化的时候被调用，我们可以通过 `this.options` 来获得部件的各项配置。

它的构造方法是这样的：

```
$.widget("mobile.dynamicimage", $.mobile.widget, {
    options: {
        width: 100,
        margin: 0
    },
    _create: function() {
        // 调用一个私有方法
        this._loadURL();
    }
});
```

每个公有方法（方法名不以下划线开头的）都可以通过部件的接口调用。通常来说，我们至少需要实现 `refresh`、`enable` 和 `disable` 这三个 jQuery Mobile 里的通用方法。举个例子，程序通过 JavaScript 改变图片的 URL 后，需要用 `$("#ourImage").dynamicimage("refresh")` 调用公有方法 `refresh` 来触发部件的刷新行为。

`refresh`、`enable`、`disable` 这三个方法的定义如下：

```
$.widget("mobile.dynamicimage", $.mobile.widget, {
    options: {
        width: 100,
        margin: 0
    },
    _create: function() {
        //调用一个私有方法
        this._loadURL();
    },
    // 公有方法
    enable: function() {
        // 启用部件
        $(this.element).attr('disabled', '');
    },
```

```
        disable: function() {
            // 禁用部件
            $(this.element).removeAttr('disabled');
        },
        refresh: function() {
            // 刷新部件
            this._loadURL();
        }
    });
```

最后是包含核心逻辑的私有方法 _loadURL 的定义：

```
    _loadURL: function() {
            // this.element 就是我们的 img 元素

            var url; // 用来定义 Sencha IO Src 服务的 URL
            url = "http://src.sencha.io/";

            var parameters = "";
            if (!isNaN(this.options.width)) {
                    parameters += "x" + this.options.width;
            }
            if ((!isNaN(this.options.margin)) && this.options.margin>0) {
                    parameters += "-" + this.options.margin;
            }
            if (parameters.length>0) {
                    url += parameters + "/";
            }

            // Sencha IO 需要提供 URL 的绝对路径
            var originalUrl = $(this.element).jqmData("src");
            if (originalUrl.length>1) {
                    var newUrl = "";
                    if ($.mobile.path.isAbsoluteUrl(originalUrl)) {
                            // 图片的 URL 是绝对路径
                            newUrl = originalUrl;
                    } else {
                            // 图片 URL 是相对路径， 我们来计算它的绝对路径
                            var baseUrl = $.mobile.path.parseUrl(location.href);
                            var baseUrlWithoutScript = baseUrl.protocol + "//" +
                                baseUrl.host + baseUrl.directory;
                            newUrl = $.mobile.path.makeUrlAbsolute(originalUrl,
                                baseUrlWithoutScript);
                    }

                    url += newUrl;

                    $(this.element).attr("src", url);
            }
```

3. 自动初始化

加上下面这段自动初始化的代码，页面就会查找所有带 date-role="dynamicimage" 属性的元素来创建 dynamicimage 实例。

```
$(document).bind("pagecreate", function(event) {
    // 找到相应的 data-role，调用 dynamicimage 的构造器
    $(event.target).find("img:jqmData(role='dynamic-image')").dynamicimage();
});
```



使用 jqmData 而不是 attr，可以使代码兼容自定义命名空间。元素中所有的 data-* 属性都会自动地匹配到部件对象内部的 this.options 中。

4. 使用插件

首先是添加插件的 JS 文件，在引入 jQuery Mobile 的 JS 文件的后面，加上：

```
<script src="jquery.mobile-dynamicimage-1.0.js"></script>
```

然后我们只需要创建配置了正确参数的 img 元素就可以了：

```
<!-- 占据屏幕全部宽度的图片 -->
<img data-src="images/photo.png" data-role="dynamic-image">

<!-- 占据屏幕 40% 宽度的图片 -->
<img data-src="images/photo.png" data-role="dynamic-image" data-width="40">

<!-- 占据屏幕 100% 宽度并且带 20 个像素的外边距的图片 -->
<img data-src="images/photo.png" data-role="dynamic-image" data-margin="50">
```

插件效果见图 10-1。



图 10-1：插件在 jQuery Mobile 页面上的运行效果，同一张图片经由 Sencha IO Src 的服务调整成不同的尺寸

5. 完整的代码

下面是 jquery.mobile-dynamicimage-1.0.js 源文件中的全部代码：

```
(function($){
     // 小部件的定义
     $.widget( "mobile.dynamicimage", $.mobile.widget, {
            options: {
                   margin: 0, width: 100
            },
            _create: function() {
                   this._loadURL();
            },

                // 私有方法
                _loadURL: function() {
                     //this.element 就是我们的 img 元素

                      var url; // 用来定义 Sencha IO Src 服务的 URL
                      url = "http://src.sencha.io/";

                      var parameters = "";
                      if (!isNaN(this.options.width)) {
                              parameters += "x" + this.options.width;
                      }
                      if ((!isNaN(this.options.margin)) && this.options.
                              margin>0) {parameters += "-" + this.options.
                              margin;
                      }
                      if (parameters.length>0) {
                              url += parameters + "/";
                      }

                      // Sencha IO 需要提供 URL 的绝对路径
                      var originalUrl = $(this.element).jqmData("src");
                      if (originalUrl.length>1) {
                          var newUrl = "";
                          if ($.mobile.path.isAbsoluteUrl(originalUrl)) {
                                  // 图片的 URL 是绝对路径
                                  newUrl = originalUrl;
                          } else {
                                  // 图片 URL 是相对路径， 我们来计算它的绝对路径
                                  var baseUrl = $.mobile.path.parseUrl
                                      (location.href);
                                  var baseUrlWithoutScript = baseUrl.
                                      protocol + "//" + baseUrl.host +
                                      baseUrl.directory;
                                  newUrl = $.mobile.path.makeUrlAbsolute
                                      (originalUrl, baseUrlWithoutScript);
                          }

                          url += newUrl;

                          $(this.element).attr("src", url);
```

```
                    }
                },

                // 公有方法
            enable: function() {
                // 启用小部件
                        $(this.element).attr('disabled', '');
            },
            disable: function() {
                // 禁用小部件
                        $(this.element).removeAttr('disabled');
            },
            refresh: function() {
                // 刷新小部件
                        this._loadURL();
            }
    });
    // 小部件定义结束

    // 自动初始化代码
    $(document).bind("pagecreate", function(event) {
        // 找到相应的 data-role，调用 dynamicimage 的构造器
        $(event.target).find("img:jqmData(role='dynamic-image')").dynamicimage();

    });

}(jQuery));
```

10.2　插件精萃

有不少很酷的插件，下面列举了一些在日常项目中最实用的插件。

10.2.1　分页插件

http://filamentgroup.com/lab/jquery_mobile_pagination_plugin 上的分页插件（图 10-2）可以用来对内容（比如图片）进行分页，它在屏幕上显示左右两个箭头，用户一看就知道可以通过箭头进行前后翻页。

用户翻页的方式包括：

- 点击浮动在屏幕上的箭头按钮；
- 使用设备键盘上的左右箭头按键；
- 通过手指在屏幕上左右轻扫。

下载插件时会看到两个文件：

- jquery.mobile.pagination.css；
- jquery.mobile.pagination.js。

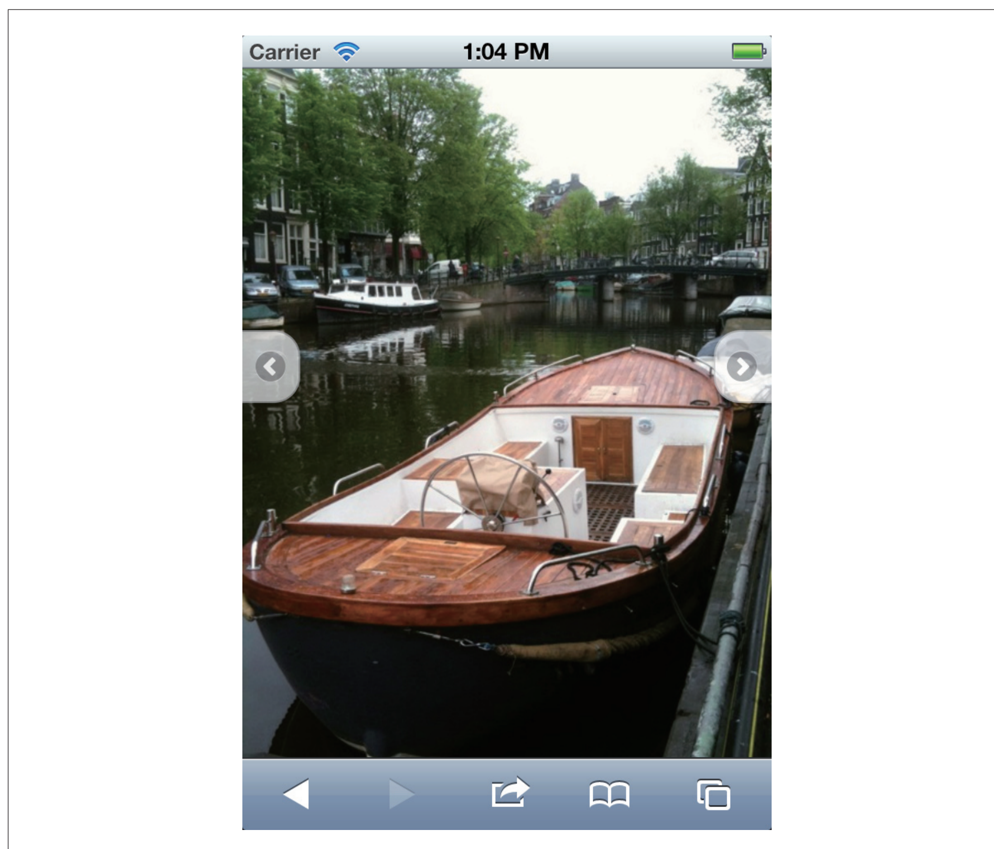


图 10-2：分页插件把一组图片分成了多个页面进行显示

把包含这两个文件的声明加入到 head 元素中之后，需要创建一个 data-role="pagination" 的 ul 元素。每个 jQuery Mobile 页面都应该包含一个分页小部件。

每个分页小部件都包含两个分别链接到上一页和下一页的 li 元素。li 元素可以用两个类来修饰：ui-pagination-prev 用来修饰链到上一页 li 元素，ui-pagination-next 用来修饰链到下一页的 li 元素。

```
<ul data-role="pagination">
   <li class="ui-pagination-prev"><a href="1.html">Prev</a></li>
   <li class="ui-pagination-next"><a href="3.html">Next</a></li>
</ul>
```

10.2.2　Bartender插件

这个插件提供了一个类似于 iOS 应用程序（图 10-3）底部的标签导航，它是免费的，可以从 http://www.stokkers.mobi/valuables/bartender.html 获得。

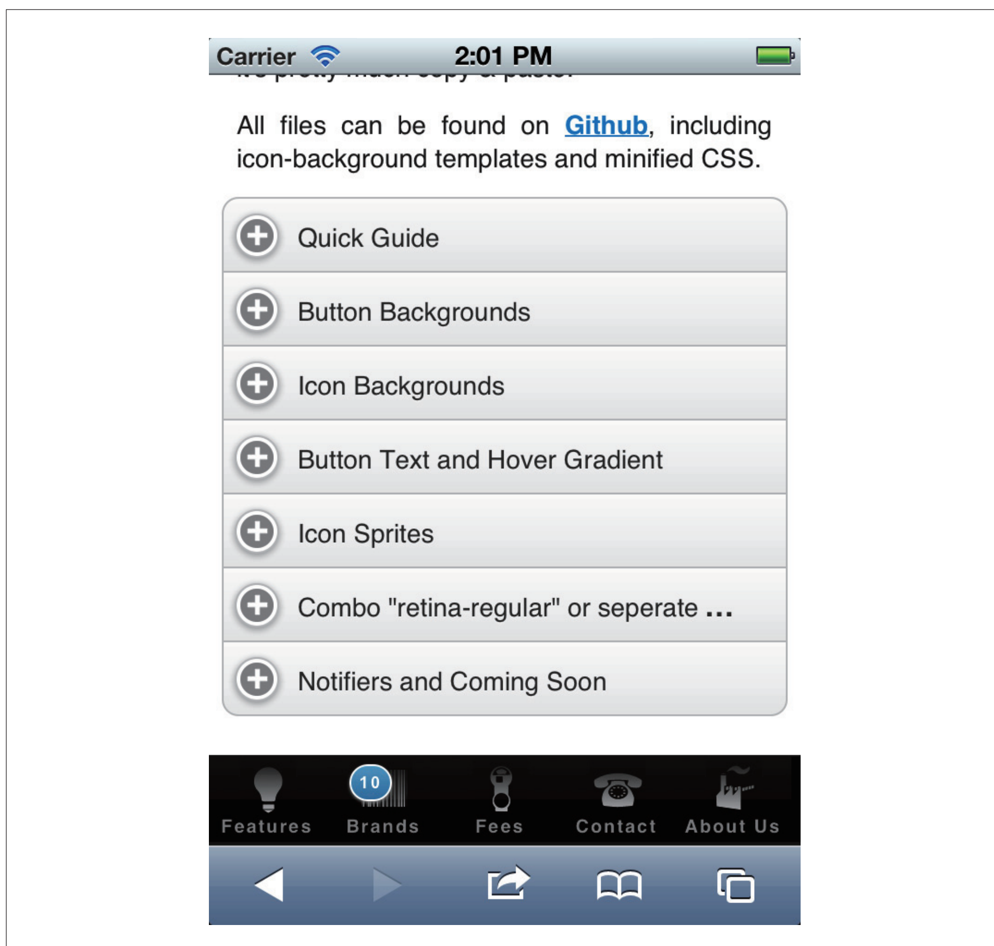


图 10-3：Bartender 插件提供了 iOS 风格的标签导航条设计

这个插件只有一个 CSS 文件和一堆图片，因此不需要用到 data-* 元素，只有一些类的定义。

模板代码通常都放在固定页脚内的 navbar 部件里，就像下面这样：

```
<div data-role="navbar" data-grid="d">
   <ul class="apple-navbar-ui comboSprite">
      <-- 元素 -->
   </ul>
</div>
```

这个插件也支持显示计数气泡的标签，和在 iOS 上一样，在 li 元素中使用 XXXX 即可。

10.2.3 DateBox插件

DateBox 提供了一个使用 jQuery Mobile 语法实现的日期选择器（图 10-4）。它是通过声明 `data-role="datebox"` 的 `<input type="date">` 元素来实现的，支持通过众多的配置项进行自定义，它的标准定义是这样的：

```
<input type="date" data-role="datebox">
```

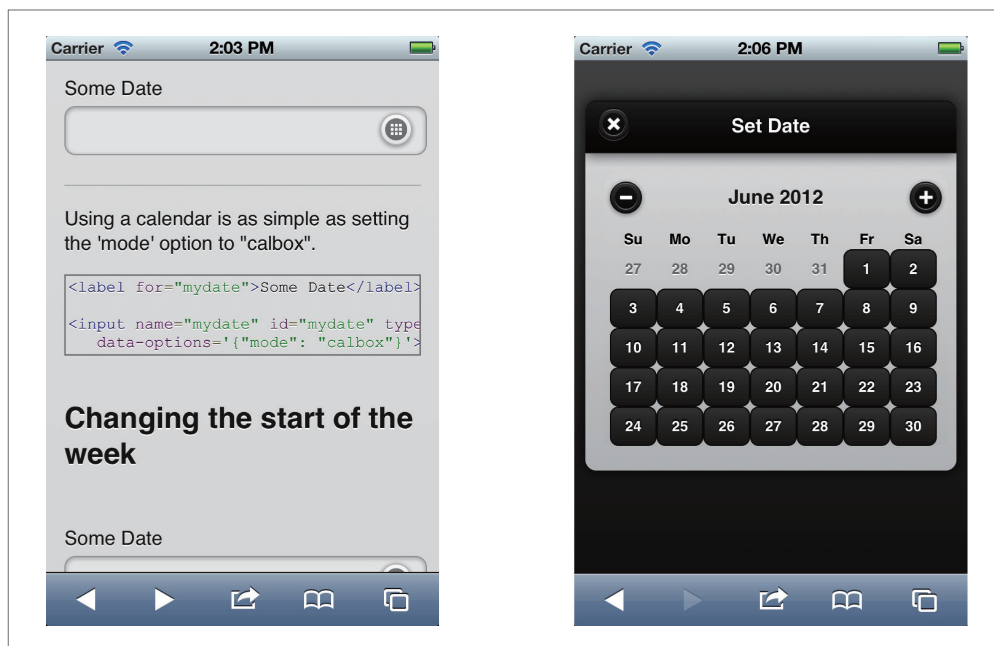


图 10-4：使用 DateBox 插件可以创建漂亮的日期选择器，就算是在不支持 HTML5 `<input type="date" />` 元素的设备上也能使用

为这个插件提供下载、示例和文档的官方网站是 http://dev.jtsage.com/jQM-DateBox，其中提供了多种模式的日期选择器，它们的相似之处在于总是放在一个弹出层内来显示日期选择（图 10-5）。这个插件支持的模式包括：

- 全日历的选择器；
- Android 模式，为年月日的设置分别提供一个数字步进器的日历；
- Slider 模式，为年月日的设置各提供一个水平滑动条；
- Flip 模式，为年月日的设置分别提供一个垂直的滑动条；
- 时间模式；
- 持续时间模式。

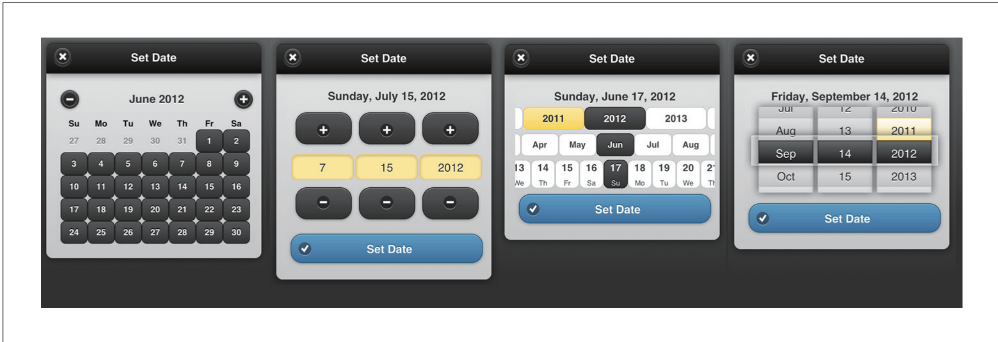


图 10-5：DateBox 插件提供了各种模式的日历选择

10.2.4　Simple Dialog插件

Simple Dialog 插件用 jQuery Mobile 风格的窗口替代标准的 `window.alert`、`window.confirm`、`window.prompt` 来获得用户的输入（图 10-6），它同样是免费的。

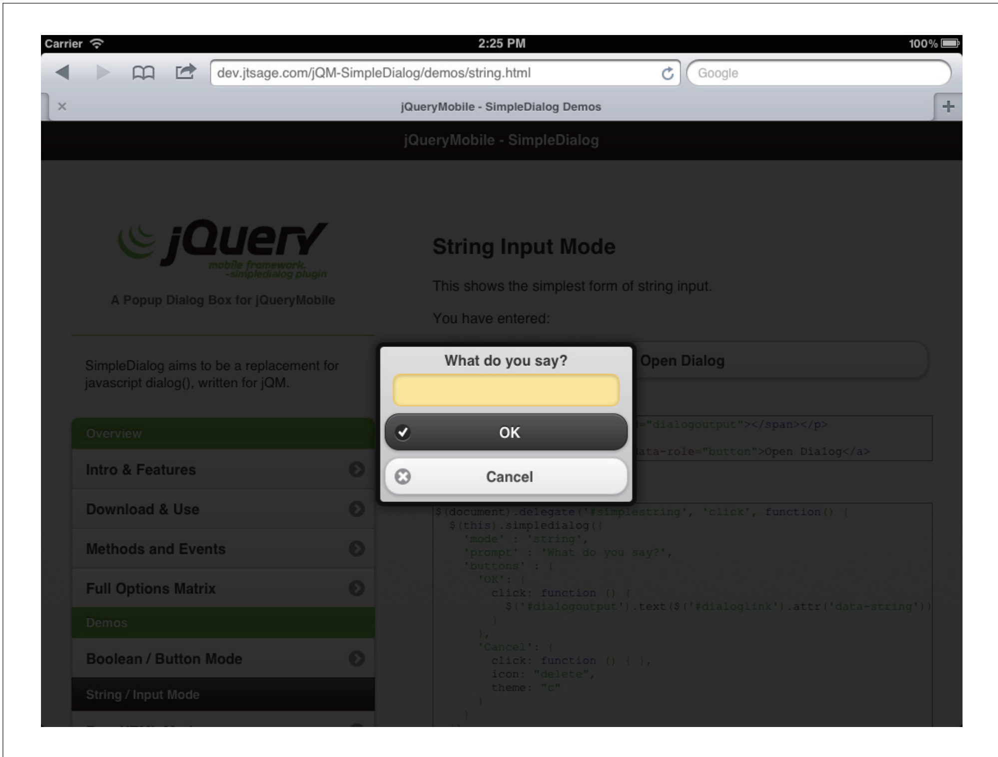


图 10-6：Simple Dialog 用 jQuery Mobile 风格的窗口替代标准的 JavaScript 弹窗

这个插件的文档和下载可以在 http://dev.jtsage.com/jQM-SimpleDialog 找到。

要使用 Simple Dialog 插件，需要下载插件的 JS 文件并把它包含到 `head` 元素中，或者使用 CDN，就像这样：

```
<script src="http://dev.jtsage.com/cdn/simpledialog/latest/jquery.mobile.
    simpledialog.min.js"></script>
```

当需要创建 `alert` 框时，可以使用下面这个代码段：

```
$("#button").click(function() {
     $(this).simpledialog({
         mode: 'bool',  // 用于 alert 或者 confirm 弹窗
         prompt: "无法打开这个文件",
         useModal: true,
         buttons: [
             'Ok':  {
                theme: "c", icon: "check"
             }
         ]
     });
});
```

如果是要创建一个 `confirmation` 对话框，可以这样：

```
$("#button").click(function() {
    $(this).simpledialog({
        mode: 'bool',
        prompt: "要删除这个文件吗？",
        useModal: true,
        buttons: [
            'Yes':  {
               theme: "c", icon: "delete",
               click: function() {
                     // 删除
               }
            },
            'No':  {
               theme: "a", icon: "cancel"
            },
        ]
    });
});
```

最后，要创建 `prompt` 提示框就这样：

```
$("#button").click(function() {
     $(this).simpledialog({
         mode: 'string',
         prompt: "你叫什么名字？",
         useModal: true,
         buttons: [
```

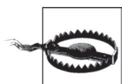
```
                'No': {
                    theme: "c", icon: "delete",
                    click: function() {
                        alert("Your name is " + $("#button").jqmData("string");
                    }
                }
            ]
        });
    });
```

10.2.5 Action Sheet插件

Action Sheet 是一个模态的弹出菜单，也是一个从 iOS 原生控件上获得启发的插件。

我们可以在 https://github.com/hiroprotagonist/jquery.mobile.actionsheet 上找到这个插件，包括 CSS 和 JavaScript 文件。

要创建一个 Action Sheet，需要一个触发它的按钮，这是一个 `data-role="actionsheet"` 的 `a` 元素。点击后，会打开这个元素的兄弟节点，或者另一个 `id` 为 `data-sheet` 属性指定 `id` 的元素。



要使用这个插件，别忘了把 jquery.mobile.actionsheet.js 和 jquery.mobile.actionsheet.css 添加到 `head` 元素中。

这个元素里面多半包含 jQuery Mobile 按钮。如果用户点击弹出块之外的区域，它就会被关闭。我们也可以在里面添加一个 `data-rel="close"` 的按钮，那么它就会执行关闭 / 取消的功能。示例代码如下：

```
<a data-role="actionsheet" data-sheet="share" data-icon="plus">Share</a>

<div id="share">
   <a href="facebook.html" data-role="button">Share in Facebook</a>
   <a href="twitter.html" data-role="button">Share in Twitter</a>
   <a href="google.html" data-role="button">Share in Google+</a>
   <a data-rel="close" data-role="button">Cancel</a>
</div>
```

10.3 供平板使用的插件

在为平板电脑开发 Web 应用时，将屏幕分成两列而不是像标准 jQuery Mobile 应用那样全屏展示是一个很实用的功能。这就是为什么 SplitView 和 MultiView 这两个插件要为平板应用提供各自解决方案的原因。

10.3.1 SplitView插件

SplitView 插件可以在 http://asyraf9.github.com/jquery-mobile 找到，利用它可以把页面分成两块区域，这两个区域也叫做面板。

每个面板都是一个 jQuery Mobile 页面，包含页头、内容区和页尾。SplitView 允许屏幕在横屏状态下同时出现两个面板（图 10-7）。那样我们就有了两个页面，通常左边的是菜单，右边是内容区域。

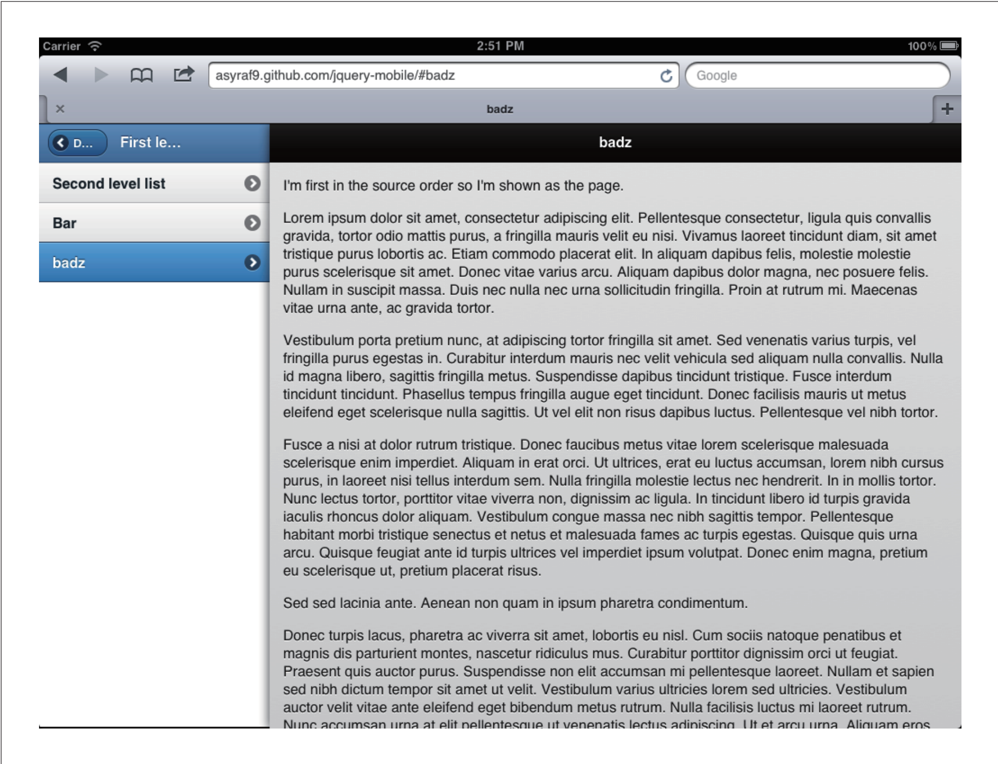


图 10-7：使用 SplitView 插件，就可以创建支持同时显示多个页面（通常情况下显示为菜单和内容区域）的平板应用程序

在竖屏状态下，菜单面板隐藏在左上角的按钮内，点击后才会以弹出菜单的形式显示出来（图 10-8）

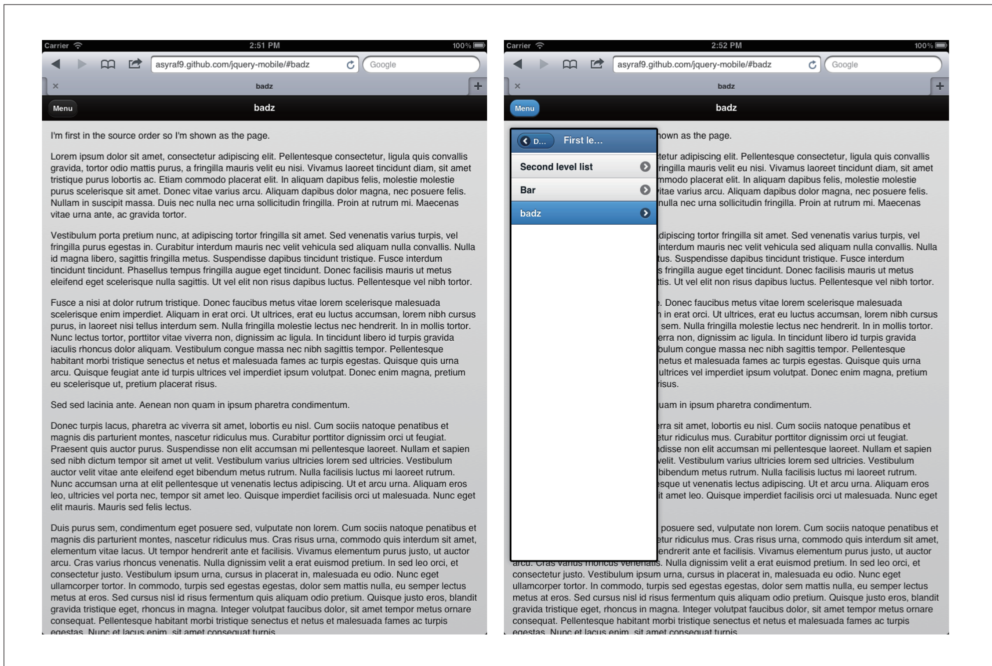


图 10-8：旋转平板到竖屏显示状态下，SplitView 和 MultiView 会调整双栏显示变为只显示主内容区，而左栏会隐藏到左上角以弹出菜单的方式显示

每个面板都有一个 `data-id` 属性用来定义面板的名字，它可以使链接变得更容易：

```
<body>
  <div data-role="panel" data-id="menu">
    <div data-role="page">

    </div>
  </div>
  <div data-role="panel" data-id="main">
    <div data-role="page">

    </div>
  </div>
</body>
```

然后，可以定义链接元素在哪个面板中加载。过渡的动画和 `frame` 一样显示在面板之内。

举个例子：

```
<a href="demo.html" data-panel="main">Demos</a>
```

10.3.2　MultiView插件

MultiView 和 SplitView 的功能差不多，但是实现却不同（图 10-9）。它也有定义 `data-role="panel"` 的元素，但是和 `page` 元素的关系却是子节点与父节点的关系。

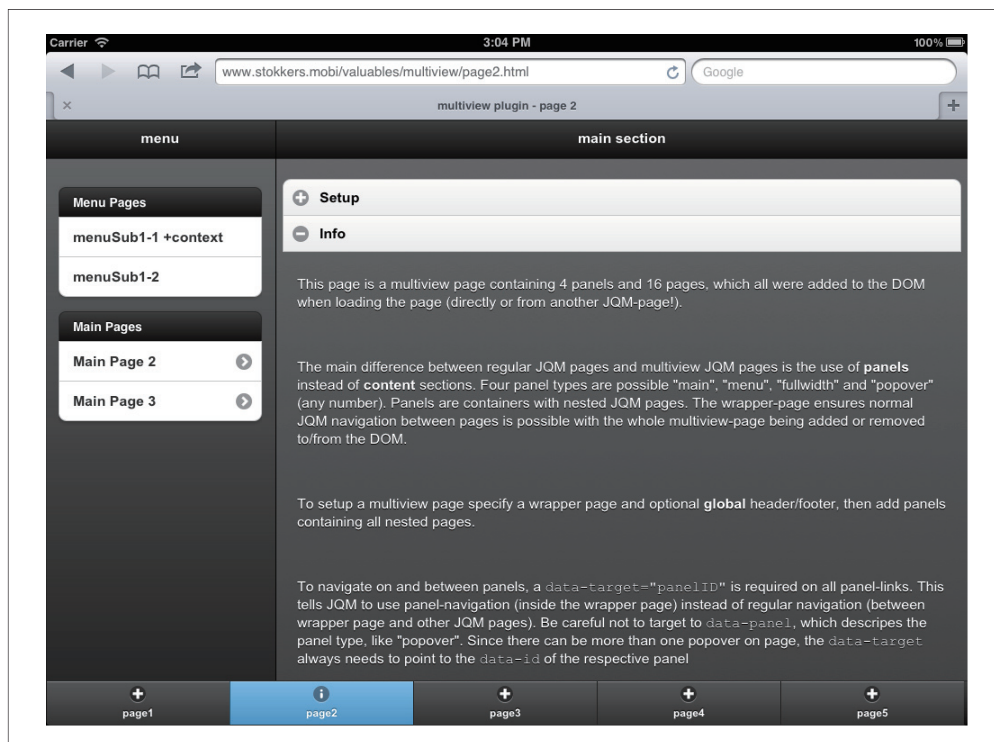


图 10-9：使用 MultiView 插件，也可以创建各面板间共享固定的页脚的平板风格的应用

http://www.stokkers.mobi/valuables/multiview/page1.html 上有这个插件的演示，它改变了 jQuery Mobile 常见的交互方式。一个 `data-role="page"` 的元素（而不是内容区域），可以包含最多四个面板区域（`data-role="panel"`）：

- 菜单面板；
- 主面板；
- 全屏面板（可选，横屏时可用）；
- 弹出面板（可选，竖屏时可用）。

面板的类型由 `data-id` 来定义，比如：`data-id="menu"`。

每个面板都包含它自己的页面，因此会存在页面嵌套的情况。它的结构是：页面➤页面➤页面➤页头 / 内容区 / 页脚。只要需要，每一个面板可以包含任意多的页面，

但至少得有一个页面具有属性 `data-show="first"`。

使用 `data-target` 属性，可以定义链接在哪里打开，它可以是：

- 全屏加载（移除所有面板）；
- 面板加载（菜单面板或者主面板）；
- 多面板加载（菜单和主面板同时加载）。

这个插件的文档不多，但是参考在线演示，要想搞明白它并不是很困难。

10.4 兼容的jQuery UI插件

有一些 jQuery UI 的插件也可以在 jQuery Mobile 应用中使用，但由于这些插件没有遵循 jQuery Mobile 框架的规范，因此不能使用 `data-role` 和一些完全由 JavaScript 代码调用提供的功能。

下面列出了一些兼容的 jQuery UI 插件。

- Photoswipe（http://photoswipe.com）：支持在 iOS、Android 和 BlackBerry 6+ 创建照片画廊的插件。
- Diapo（http://www.pixedelic.com/plugins/diapo）：有很棒的 CSS 动画效果的幻灯片画廊。
- jQuery UI Maps（http://code.google.com/p/jquery-ui-map）：在移动 Web 应用中集成 Google 地图的插件。
- MobiScroll（http://www.mobiscroll.com）：使用步进器或者轮盘的方式进行时间日期的选择。

第 11 章

为应用商店打包

有人说，应用商店代表了应用的未来。说实话，没人知道未来的应用发布方式究竟是基于应用商店、基于浏览器，还是两者混合的方式。使用 HTML5（特别是 jQuery Mobile）的绝妙之处在于，我们既可以基于浏览器来发布应用，也可以打包成原生应用通过应用商店发布。

因此，如果想通过应用商店来发布应用，使用 jQuery Mobile 来开发绝对是没有问题的。不过，要做到这一点，针对不同的目标平台有不同的方式。

首先要明白的一点是不同的平台需要创建不同的软件包。从某种意义上来说，我们需要把所有的文件（HTML、JavaScript、CSS 和 jQuery Mobile 框架的文件）拷贝到不同的项目中，然后创建不同的软件包。



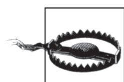
W3C 有一个社区工作组正在起草用于应用发布的打包标准，就是 Native Web Apps，它的官方地址是 http://www.w3.org/community/native-web-apps/。

把 Web 应用当作原生应用来打包，应用可以调用到一些非 HTML5 的 API，这些 API 包括相机、联系人列表、加速度传感器 API。

要打包应用到商店发布，可以选择了下方式。

- 为每个平台创建一个原生应用项目，把 Web 应用的文件作为本地资源加入到项目中，用 Web View 组件绑定应用的 HTML 内容。这种方式有时被叫做混合应用。

• 使用某个官方的 Web 应用平台，这时往往会把我们的文件打包成一个 zip 压缩包。
• 使用原生应用编译工具，帮助我们为各个平台编译相应的软件包。



把 Web 应用编译成原生软件包往往需要掌握各平台原生应用代码和 SDK 工具的专业知识。

11.1 到应用商店去发布

打包应用的第一步是明确应用要针对什么平台在哪个应用商店进行发布。

表 11-1 列举了可以用来发布应用的商店，我们需要在各个平台上都创建一个应用发布账号。

表11-1：可以发布应用的商店列表

商　店	所　有　者	平　台	文件格式	发布费用	URL
AppStore	Apple	iOS（iPhone、iPod、iPad）	ipa	每年 99 美元	http://developer.apple.com/programs/ios
Android Market	Google	Android	apk	一次 20 美元	http://market.android.com/publish
AppWorld	RIM BlackBerry	Smartphones/PlayBook	cod/bar	免费	http://appworld.blackberry.com/isvportal
Nokia Store	Nokia	Symbian/N9	wgz/deb bar	1 欧元	http://info.publish.nokia.com/
Amazon AppStore	Amazon	Android/Kindle Fire	apk	每年 99 美元	http://developer.amazon.com/
MarketPlace	Microsoft	Windows Phone		每年 99 美元	http://create.msdn.com/

针对每个平台，有必要根据各自的文档检查一遍各个应用商店还需要提供哪些元数据，比如：

• 高分辨率的图标（通常都是 512 × 512）；
• 应用描述；
• 所属分类的选择；
• 每个平台上的应用截图；
• 发布 – 兼容设备列表；
• 发布 – 国家和语言；
• 市场营销口号 。

11.2 自定义的发布

某些平台允许我们通过自己的 Web 服务器发布应用，前提是确保 MIME 类型配置正确。Symbian 和 BlackBerry 设备可以毫无障碍地接收和安装任何 Web 应用。

对于 Android 和 Nokia N9 设备，只有用户在设置中允许从第三方服务器进行安装后，才能够从我们的服务器上安装应用。

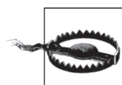
其他平台，比如 iOS 或者 Windows Phone，则要求必须从官方应用商店进行安装。



也有方法可以对设备进行配置，使得它可以不通过应用商店直接安装应用，但只能用作测试，这些技术不在本书的讨论范围之内。

11.3 准备打包

有必要时刻提醒自己，我们是在用 jQuery Mobile 创建原生应用风格的 Web 应用。因此，第一个要注意的就是浏览器工具条没有了。举个例子来说，界面上不再默认包含后退按钮了。应用必须要在页头创建一个后退按钮，就像我们在前面介绍的一样。



编译原生应用时，切记不要通过 CDN 的方式来加载 jQuery Mobile 文件。

第二个值得留心的问题和加载外部页面有关。在创建应用时，我们可以加载本地的随应用一起发布的 HTML 页面。

如果需要从一个外部的 URL 来加载 jQuery Mobile 的文档，我们可以在 `mobileinit` 事件的处理程序中设置 `$.support.cors=true`，允许框架使用 AJAX 访问远程服务器。同时还需要设置 `$.mobile.allowCrossDomainPage=true`。

jQuery Mobile 建议关闭 `pushState`，以避免和 URL 管理产生冲突。

因此，我们的 HTML 入口页面里应该包含这样一段代码：

```
$(document).bind("mobileinit", function() {
    $.support.cors = true;
    $.mobile.allowCrossDomainPages = true;
    $.mobile.pushState = false;
});
```

如果应用是面向 iOS 5.0 以上的版本，那就可以在固定的头部和底部使用原生的固定工具条，这样能获得更好的性能和可用性。



在打包原生应用时，还需要准备其他一些文件，比如用在主屏幕或者应用菜单上的图标，应用启动时需要展示的启动图片。一些平台还要求提供一个应用可以访问的网络服务器的列表。

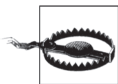
11.4 使用PhoneGap打包

PhoneGap 是一个将 HTML5 代码编译成原生应用的开源平台，它完全兼容 jQuery Mobile。PhoneGap（之前也叫做 Apache CallBack）主要由 Adobe 和另外几家颇具影响力的公司维护。完整地解释 PhoneGap 超出了本书的范围，这里只是简单介绍些入门知识。

使用 PhoneGap（http://phonegap.com），可以把 jQuery Mobile 应用编译成下列平台的应用：

- iOS；
- Android；
- webOS；
- Symbian；
- BlackBerry；
- Windows Phone。

下载 PhoneGap 后，可以得到一个 ZIP 压缩包，包含了所有平台下使用各自 SDK 打包的示例，以及一个需要包含到所有 HTML 文件中去的 JS 文件。直到 PhoneGap 1.2，每一个平台都还需要一个自己的 JS 文件。这个文件用于保持各平台部分行为一致，并加强 API 兼容性。



如果要编译 iOS 应用，需要一台 Mac 或者一个提供 Mac 环境的云服务，在 http://macincloud.com 可方便地租到一个 Mac 设备。

在 iOS 上，PhoneGap 提供了一个安装方法，可以把 PhoneGap 项目类型加入到 Xcode 中。Xcode 是创建原生 iOS 应用的官方 IDE。表 11-2 列出了各平台下编译应用所必须安装的 SDK。

表11-2：编译各平台原生应用所需要用到的SDK和IDE

SDK	设备平台	桌面平台	下载地址
Xcode	iOS	MacOS	通过 Mac AppStore 下载
ADT for Eclipse	Android	Win/Mac/Linux	http://developer.android.com
WebWorks SDK	BlackBerry	Win/Mac	http:// blackberry.com/developers

（续）

SDK	设备平台	桌面平台	下载地址
VisualStudio for WP	Windows Phone	Win	http://microsoft.com/visualstudio
Nokia Web Tools for Symbian	Symbian	Win/Mac	http://developer.nokia.com

如果要为各个平台都创建一个项目，需要把自己的文件（HTML、JavaScript、CSS、图片等）复制到每个 PhoneGap 示例项目对应的目录下。通常这是一个叫做 www 的目录。找到一个叫做 index.html 的文件，那就是需要用我们自己的 jQuery Mobile 应用替换的文件。

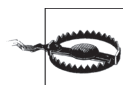
PhoneGap Build

如果你嫌为每个平台使用各自的 SDK 创建一个包麻烦，Adobe 倒是提供了一个在云端编译应用的服务（既有免费的方案也有付费的方案），叫做 PhoneGap Build，它的地址是 http://build.phonegap.com。

使用这个服务，我们可以把完整版的 jQuery Mobile 应用（包括 CSS、JavaScript 和图片）和一个遵循 W3C 小部件规范的 config.xml 文件打成一个 ZIP 压缩包上传，PhoneGap Build 就会把它编译成下列平台的原生软件包：

- iOS；
- Android；
- webOS；
- BlackBerry 智能手机；
- Symbian。

在写作本书的时候，PhoneGap Build 还不支持 Windows Phone 和 BlackBerry 平板电脑，但应该很快就会支持了。



对于 iOS 和 BlackBerry，我们需要把平台发布程序提供的签名私钥提供给 PhoneGap Build。

关于封面

本书封面上的动物是日本黑貂（属于紫貂），主要生活在日本最北部的北海道岛上，是生活在俄罗斯、蒙古，以及中国北方针叶林中的紫貂的一种。

黑貂柔软、光滑的貂皮很值钱。自中世纪以来，一直是兽皮贸易中的主要商品，深受早期欧洲皇室的青睐。俄罗斯产的貂皮最为贵重。今天，这些貂皮既有来自商业养殖的，也有来自野生捕杀的。

黑貂身长约 40 厘米，这不包括 10 到 20 厘米长的尾巴，体重 900 到 2000 克上下。它们的皮有从浅棕色到深棕色的过渡，而且还带有浅色斑点。日本的这一种腿和脚上有黑色印记。

这些杂食的哺乳动物喜昏暗（一般黄昏时分活动），它们吃动物、鸟类、鱼、蛋、昆虫和植物。它们还能捕食体型比自己大的动物，甚至包括小鹿，有时候也吃其他动物剩下的猎物。黑貂一窝生三个幼崽，一般怀孕一个月，延迟着床八个月（胚胎在此期间处于休眠状态）。有时候，黑貂会与松貂异种交配，生下的杂交品种叫 kidus。

本书封面动物原载于 *Wood's Animate Creation*。

jQuery Mobile即学即用

你想写一个Web应用，让它既能在iPad和Kindle Fire上，也能在iPhone和Android手机上运行吗？这本介绍jQuery Mobile的书会告诉你怎么做。通过一系列手把手的练习，你能够学会以最有效的方式使用这个框架的各种界面组件，构建定制的、跨平台的应用。即使没有太多编程经验，不熟悉jQuery也不要紧，本书很容易看懂。

看完这本书之后，你就会知道怎么使用jQuery Mobile和语义化HTML5代码，创建出能在各种智能手机和平板电脑上运行的响应式、基于AJAX的界面。

- 理解jQuery Mobile与HTML5、CSS3以及JavaScript的关系
- 利用UI组件创建表单、列表、导航条和按钮
- 使用JavaScript、AJAX和jQuery核心框架创建动态内容
- 通过主题和CSS3完全定制用户界面
- 让用户能够从浏览器中安装应用并离线使用
- 打包成原生应用并通过应用商店发售

“《jQuery Mobile即学即用》通俗易懂地介绍了jQuery Mobile框架，旨在帮读者尽快成为一名高效开发者。”

—— appendTo（http://appendto.com/）首席执行官Mike Hostetler

Maximiliano Firtman是移动开发和HTML5技术专家。身为Web和移动技术教育工作者，他既是一位培训师，又是一位演讲家，还是Adobe社区专家和*Programming the Mobile Web*（O'Reilly）一书的作者。另外，他还是ITMaster专业培训机构（http://www.itmaster.com.ar/）的创始人。



封面设计：Karen Montgomery　马冬燕
图灵社区：www.ituring.com.cn
新浪微博：@图灵教育 @图灵社区
反馈/投稿/推荐信箱：contact@turingbook.com
热线：(010)51095186转604

分类建议　计算机/移动开发

人民邮电出版社网址：www.ptpress.com.cn

O'Reilly Media, Inc.授权人民邮电出版社出版

O'REILLY®
oreilly.com



ISBN 978-7-115-30294-6
定价：49.00元

欢迎加入

图灵社区

最前沿的IT类电子书发售平台

电子出版的时代已经来临。在许多出版界同行还在犹豫彷徨的时候，图灵社区已经采取实际行动拥抱这个出版业巨变。作为国内第一家发售电子图书的IT类出版商，图灵社区目前为读者提供两种DRM-free的阅读体验：在线阅读和PDF。

相比纸质书，电子书具有许多明显的优势。它不仅发布快，更新容易，而且尽可能采用了彩色图片（即使有的书纸质版是黑白印刷的）。读者还可以方便地进行搜索、剪贴、复制和打印。

图灵社区进一步把传统出版流程与电子书出版业务紧密结合，目前已实现作译者网上交稿、编辑网上审稿、按章发布的电子出版模式。这种新的出版模式，我们称之为“敏捷出版”，它可以让读者以较快的速度了解到国外最新技术图书的内容，弥补以往翻译版技术书“出版即过时”的缺憾。同时，敏捷出版使得作、译、编、读的交流更为方便，可以提前消灭书稿中的错误，最大程度地保证图书出版的质量。

现在购买电子书,读者将获赠书款20%的社区银子,可用于兑换纸质样书。

最方便的开放出版平台

图灵社区向读者开放在线写作功能，协助你实现自出版和开源出版的梦想。利用“合集”功能，你就能联合二三好友共同创作一部技术参考书，以免费或收费的形式提供给读者。（收费形式须经过图灵社区立项评审。）这极大地降低了出版的门槛。只要你有写作的意愿，图灵社区就能帮助你实现这个梦想。成熟的书稿，有机会入选出版计划，同时出版纸质书。

图灵社区引进出版的外文图书，都将在立项后马上在社区公布。如果你有意翻译哪本图书，欢迎你来社区申请。只要你通过试译的考验，即可签约成为图灵的译者。当然，要想成功地完成一本书的翻译工作，是需要有坚强的毅力的。

最直接的读者交流平台

在图灵社区，你可以十分方便地写作文章、提交勘误、发表评论，以各种方式与作译者、编辑人员和其他读者进行交流互动。提交勘误还能够获赠社区银子。

你可以积极参与社区经常开展的访谈、审读、评选等多种活动，赢取积分和银子，积累个人声望。

ituring.com.cn